Extra Domain Data Generation with Generative Adversarial Nets

Luuk Boulogne
Bernoulli Institute
Department of Artificial Intelligence
University of Groningen
Groningen, The Netherlands
lhboulogne@gmail.com

Klaas Dijkstra
Centre of Expertise in
Computer Vision & Data Science
NHL Stenden University of Applied Sciences
Leeuwarden, The Netherlands
k.dijkstra@nhl.nl

Marco Wiering
Bernoulli Institute
Department of Artificial Intelligence
University of Groningen
Groningen, The Netherlands
m.a.wiering@rug.nl

***Abstract*—This study focuses on supplementing data sets with data of absent classes by using other, similar data sets in which these classes are represented. The data is generated using Generative Adversarial Nets (GANs) trained on the CelebA and MNIST datasets. In particular we use and compare Coupled GANs (CoGANs), Auxiliary Classifier GANs (AC-GANs) and a novel combination of the two (CoAC-GANs) to generate image data of domain-class combinations that were removed from the training data. We also train classifiers on the generated data. The results show that AC-GANs and CoAC-GANs can be used successfully to generate labeled data from domain-class combinations that are absent from the training data. Furthermore, they suggest that the preference for one of the two types of generative models depends on training set characteristics. Classifiers trained on the generated data can accurately classify unseen data from the missing domain-class combinations.**

***Index Terms*—Generative adversarial net, deep neural network, data generation**

I. Introduction

ANY image data set only covers a fixed domain. This severely limits the abilities of classifiers trained on them. E.g. classifiers are unable to classify classes that do not exist in their training sets. They also often lack in accuracy when tested on data sets different from their training set with an overlapping set of classes, since the data set specifics such as image style are rarely identical to the specifics of the training set.

Domain adaptation is the research area devoted to solve the problem of classifying data from some domain $\mathcal{C}$ where no class labels are available by using data from a different domain $\mathcal{D}$ in which labeled data are available. Similar to domain adaptation this work involves using data from some domain to expand classifier capabilities in another. We consider the setting with two domains of labeled data, where in one of the domains, for one or more classes, no data are available.

More specifically, consider image domains $\mathcal{A}$ and $\mathcal{B}$ that produce the samples in data sets A and B respectively. Furthermore samples in A have classes from set C_A and samples in B have classes from set C_B. Let there be a class c such that $c \in C_B$, but $c \notin C_A$, and furthermore $C_A \subset C_B$. From here on, the set of samples in domain $\mathcal{A}$ of class c are denoted by $\mathcal{A}^c$. For this work, we define Extra Domain Data Generation (EDDG) as the problem of generating samples from $\mathcal{A}^c$. An example of this is generating fives in the style of A if A and B contain images of digits, with B containing images of fives and A not containing images of fives.

We aim to tackle EDDG with the use of generative models (GM). GMs are able to generate new data from distributions of interest. Different types of GMs exist, such as variational auto-encoders [1] and Generative Adversarial Nets [2] (GANs) that transform noise to meaningful data. Many extensions to the GAN architecture have been made. E.g. Coupled GANs [3] (CoGANs) have shown to be useful for domain adaptation and Auxiliary Classifier GANs [4] (AC-GANs) are one of the types of GANs that are able to generate labeled data.

GANs have also been used for image-to-image translation in e.g. [5], [6]. Instead of generating new images from only noise and optionally a class label, such models alter existing images from one domain so that they seem to originate from another domain or they alter existing images of some class to appear as if they have another class label. In particular, StarGAN [6] is able to translate images in some domain to appear to have a class label for which no data in that domain is available by using additional data from another domain in which the particular class label is represented.

In this work, we do not perform image-to-image translation, but train GANs on the CelebA [7] dataset and on a combination of the MNIST [8] and MNIST-edge [3] datasets to generate original images. Primarily, we use AC-GANs and propose a novel combination of AC-GANs and CoGANs to generate data from domain-class combinations that are not present in their training sets. We also show that using the data generated by these models, classifiers can be trained that are able to accurately classify this missing data.

We first explain the GAN architectures we use in section II and then theoretically deliberate upon the usefulness of these architectures for EDDG in section III. Afterwards we discuss the experiments we have performed with them.

II. Methods

A. Generative Adversarial Nets

Generative Adversarial Nets (GANs) [2] consist of two universal function approximators, a generator and a discriminator, which are trained in a competitive setting.

The generator learns a mapping from some distribution $p_{\mathbf{z}}$ to the distribution of interest p_{data}. It gets as input a noise sample $\mathbf{z} \sim p_{\mathbf{z}}$ and outputs some fake data point $G(\mathbf{z})$. We denote the distribution of $G(\mathbf{z})$ as p_G.

The discriminator learns to predict the probability that its input $\mathbf{x}$ is sampled from the real data instead of it being produced by the generator. It learns this by maximizing its output when $\mathbf{x} \sim p_{\text{data}}$ and minimizing its output when $\mathbf{x} \sim p_G$. In contrast, the generator learns to produce samples $\mathbf{x} \sim p_G$ that maximize the output of the discriminator.

For generator G and discriminator D, the GAN objectives are captured in the value function:

$$\begin{aligned} V(D, G, p_{\text{data}}) = {} & \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] \\ & + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]. \end{aligned} \tag{1}$$

The discriminator maximizes this function, while the generator minimizes its right term. The complete training process is described by playing the following minimax game:

$$\min_G \max_D V(D, G, p_{\text{data}}).$$

The generator and discriminator in GANs are often implemented as Neural Networks (NNs) [9]. For image generation tasks, Convolutional NNs (CNNs) [10] are a popular choice, although with capsule networks [11] promising results have also been obtained [12], [13].

B. Auxiliary Classifier Generative Adversarial Nets

With Auxillary Classifier GANs [4] (AC-GANs), next to the noise vector $\mathbf{z}$, the generator receives a class label $\mathbf{c}$ as input. The discriminator does not receive $\mathbf{c}$ as input. Instead, in addition to predicting whether its input comes from the real data set, the discriminator is tasked with predicting $\mathbf{c}$.

The discriminator thus has two outputs. The first one is the vanilla GAN output $D(\mathbf{x})$, which is the probability that the source, S, from which the discriminator input, $\mathbf{x}$, is sampled is the real data set and not the distribution generated by the generator. The second output of the discriminator contains $P(C \mid \mathbf{x})$ for all classes C, which is a vector that contains the probability distribution over the class labels.

The log-likelihood for predicting the correct source, described in (1), becomes:

$$\begin{aligned} V_S = {} & \mathbb{E}_{\mathbf{x},\mathbf{c} \sim p_{\text{data}}}[\log D(\mathbf{x})] \\ & + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}, \mathbf{c} \sim p_{\mathbf{c}}}[\log(1 - D(G(\mathbf{z}, \mathbf{c})))]. \end{aligned} \tag{2}$$

The log-likelihood for predicting the correct class is formulated as:

$$\begin{aligned} V_C = {} & \mathbb{E}_{\mathbf{x},\mathbf{c} \sim p_{\text{data}}}[\log P(C = \mathbf{c} \mid \mathbf{x})] \\ & + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}, \mathbf{c} \sim p_{\mathbf{c}}},[\log P(C = \mathbf{c} \mid G(\mathbf{z}, \mathbf{c}))]. \end{aligned} \tag{3}$$

Both the generator and the discriminator are trained to maximize V_C. AC-GANs are thus trained by letting the discriminator maximize $V_C + V_S$ and letting the generator maximize $V_C - V_S$. AC-GANs are easily extended to work with multiple class labels by adding additional V_C terms to these value functions.

C. Coupled Generative Adversarial Nets

Coupled GANs (CoGANs) [3] learn to sample from a joint distribution, but are trained only using multiple marginal distributions, where each marginal distribution describes a different domain.

CoGANs are implemented as multiple GANs, each of which learns to generate samples from a different marginal distribution. The GANs are interdependent by sharing the weights in the first layer(s) of their generators and in the last layer(s) of their discriminators. When CoGANs are implemented as deep feed-forward NNs, high level image semantics are expected to be encoded in these shared weights and low level details are expected to be encoded in the weights that are not shared. This is the case because of the hierarchical way features are represented in the layers of this type of model. When a single noise vector is presented to all the GANs that make up a CoGAN, the high level semantics of the generated images will be the same, while the low level details will be different. Since each of these GANs produces images from a different domain, the tuple of images generated by presenting a single noise vector to CoGANs is the approximation of a sample of a joint distribution of these different domains.

Consider N marginal distributions $p_{\text{data}i}$ where $i \in [1..N]$. A CoGAN consisting of N GANs is trained to generate samples from the joint distribution that contains N-tuples of samples of $p_{\text{data}i}$. For each $i \in [1..N]$, GAN$_i$, with discriminator D_i and generator G_i, is trained to produce samples from $p_{\text{data}i}$. The value function for CoGANs is:

$$V_{\text{CoGAN}} = \sum_{i=1}^{N} V(D_i, G_i, p_{\text{data}i}). \tag{4}$$

D. Coupled Auxiliary Classifier Generative Adversarial Nets

By combining the value functions of CoGANs and AC-GANs, we propose Coupled Auxiliary Classifier GANs (CoAC-GANs). For the i^{th} GAN in a CoAC-GAN with a discriminator that outputs the tuple $D_i(\mathbf{x})$, $P_i(C \mid \mathbf{x})$ the log-likelihoods from (2) and (3) respectively become:

$$\begin{aligned} V_{Si} = {} & \mathbb{E}_{\mathbf{x},\mathbf{c} \sim p_{\text{data}i}}[\log D_i(\mathbf{x})] \\ & + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}, \mathbf{c} \sim p_{\mathbf{c}i}}[\log(1 - D_i(G_i(\mathbf{z}, \mathbf{c})))] \end{aligned}$$

and

$$\begin{aligned} V_{Ci} = {} & \mathbb{E}_{\mathbf{x},\mathbf{c} \sim p_{\text{data}i}}[\log P_i(C = \mathbf{c} \mid \mathbf{x})] \\ & + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}, \mathbf{c} \sim p_{\mathbf{c}i}}[\log P_i(C = \mathbf{c} \mid G_i(\mathbf{z}, \mathbf{c}))]. \end{aligned}$$

A CoAC-GAN consisting of N GANs is trained by letting the discriminators maximize:

$$\sum_{i=1}^{N} V_{Ci} + V_{Si} \tag{5}$$

and letting the generators maximize:

$$\sum_{i=1}^{N} V_{Ci} - V_{Si}. \tag{6}$$

E. *Wasserstein Generative Adversarial Nets*

Wasserstein GANs [14] (WGANs) are GANs that are trained to minimize the Earth Mover (EM) distance between p_G and p_{data}. In this optimization problem, the EM distance is defined using the Kantorovich-Rubinstein duality. This definition assumes the discriminator (named critic in the paper, since the WGAN discriminator is not trained to classify) to be K-Lipschitz [14]. Instead of forcing the discriminator to be K-Lipschitz, a soft constraint in the form of a gradient penalty can be added to the WGAN value function [15]. With this gradient penalty, the WGAN critic value function becomes:

$$\begin{aligned} V_W(D, G, p_{\text{data}}) &= \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[D(G(\mathbf{z}))] \\ &\quad - \lambda \mathbb{E}_{\hat{\mathbf{x}}\sim p_{\hat{\mathbf{x}}}}\left[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2\right]. \end{aligned}$$

The generator is trained to maximize $\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[D(G(\mathbf{z}))]$. The coefficient of the gradient penalty λ has a default value of 10. $p_{\hat{\mathbf{x}}}$ is defined sampling uniformly from lines between points from p_{data} and points from p_G.

A CoGAN trained with the WGAN objective has the value function described in (4), where $V(D_i, G_i, p_{\text{data}i})$ is substituted with $V_W(D_i, G_i, p_{\text{data}i})$. A CoAC-GAN trained with the WGAN objective has value functions for the discriminator and generator described in (5) and (6) respectively, where V_{Si} is substituted with

$$\begin{aligned} V_{WSi} &= \mathbb{E}_{\mathbf{x},\mathbf{c}\sim p_{\text{data}i}}[D_i(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}},\mathbf{c}\sim p_{\mathbf{c}i}}[D_i(G_i(\mathbf{z},\mathbf{c}))] \\ &\quad - \lambda \mathbb{E}_{\hat{\mathbf{x}}\sim p_{\hat{\mathbf{x}}i}}\left[(\|\nabla_{\hat{\mathbf{x}}} D_i(\hat{\mathbf{x}})\|_2 - 1)^2\right]. \end{aligned}$$

III. Extra Domain Data Generation

A. *Coupled Generative Adversarial Nets*

Consider some CoGAN that is trained to generate data from a joint distribution that consists of the domains $\mathcal{A}$ and $\mathcal{B}$. We denote its GAN that is trained to generate data from $\mathcal{A}$ as GAN$_{\mathcal{A}}$ and its other GAN as GAN$_{\mathcal{B}}$. Especially when classes are characterized by low level image details, their representations are expected to be encoded in the earlier shared layers. However, trying to generate images from $\mathcal{A}^c$ in this way has the following downsides:

1) Discriminators in a CoGAN are trained to only classify samples from their input distribution of real data as real and to classify all input produced by the generator as fake. Furthermore, the discriminator of GAN$_{\mathcal{A}}$ never gets a real sample from $\mathcal{A}^c$ as input during training. Therefore, for the discriminator of GAN$_{\mathcal{A}}$, fake samples generated by the generator that are similar to samples from $\mathcal{A}^c$ should be easier to reject than fake samples generated by the generator that are similar to other samples form $\mathcal{A}$. This makes it harder for the generator to fool the discriminator by producing samples from $\mathcal{A}^c$, which in turn makes it more likely for the generator to find some optimum where it is unable to generate samples from $\mathcal{A}^c$.
2) When training is done, even if GAN$_{\mathcal{A}}$ is able to generate samples from $\mathcal{A}^c$, there is no simple way to obtain them. In order to generate the desired samples from $\mathcal{A}^c$ one has to use this generator to generate samples randomly, until a sample from $\mathcal{A}^c$ is found.

B. *Auxiliary Classifier Generative Adversarial Nets*

Both issues can be eliminated by using AC-GANs for the data generation. This can be done by using both a domain label and class label as conditional input for the generator and as a variable that the discriminator has to predict. Evidently, this resolves the second issue, since after training, the generator can be primed to generate data from a specific class in a specific domain.

When training such an AC-GAN, the first issue is also taken care of when during training, the generator is presented with class labels from C_A as input when it is primed to generate data from $\mathcal{A}$ and with labels from C_B when generating data from $\mathcal{B}$. This way, the generator is never tasked to generate samples from $\mathcal{A}^c$ during training. It thus never gets to see that fooling the discriminator with data from $\mathcal{A}^c$ is difficult.

Although in this way the AC-GAN is never trained to specifically generate data from $\mathcal{A}^c$, it is trained to generate domain specific data depending on its input domain label and class specific data depending on its input class label. As long as the complexity of the function that it learns to do so is sufficiently restricted, there should be a large overlap in the features that the AC-GAN uses to generate data from different classes from the same domain. Similarly, there should be a large overlap in the features that the AC-GAN uses to generate data from the same class, but from different domains. Therefore, even though the generator never generates samples from $\mathcal{A}^c$ during training, it would still likely have the ability to generate samples of $\mathcal{A}^c$ after training.

C. *Coupled Auxiliary Classifier Generative Adversarial Nets*

As with a regular AC-GANs, because the generators of a CoAC-GAN have a conditional input variable that determines the class of the generated samples, it is possible to constrain the generator from generating samples from $\mathcal{A}^c$ during training in the same way as described in section III-B, which avoids the first downside stated in section III-A. Similarly, the second downside is also avoided, because when training is done, the class variable can be used to specify that a sample from $\mathcal{A}^c$ has to be generated using the generator of GAN$_{\mathcal{A}}$.

IV. Experiments

A. *Data sets*

Experiments were done using the 28 $\times$ 28 pixels MNIST digit data set [8], a variant of the MNIST data set where all images have been modified so that they depict the edges of the digits [3] (MNIST-edge), and the CelebA face data set [7], which consists of faces that are labeled with binary attribute labels such as ‘smiling’ and ‘male’. The CelebA images were centered and cropped to a size of 160 $\times$ 160 pixels and consecutively resized to 64 $\times$ 64 pixels. For each data set, the pixel values were normalized to lay in the domain [-1,1].

B. Extra Domain Data Generation

For data without class labels, image generation is done with CoGANs and for labeled data generation it is done with CoAC-GANs and AC-GANs. The GANs are trained on a combination of the MNIST and MNIST-edge data sets and on the CelebA data set. For the digit data set, the MNIST and MNIST-edge data set represent different domains. For the CelebA data set, the image domain is determined by gender and the classes are 'smiling' and 'not smiling'.

Three data sets with missing classes are emulated. For the face image generation task, no smiling males are presented to the discriminator during training. This data set will be denoted with Celeba^1. Similarly, for the digit generation task, some digits from the MNIST-edge are not presented during training. Experiments are performed where the digit '5' is missing and where the digits '5'-'9' are missing. These data sets are denoted MNIST^1 and MNIST^2 respectively.

GANs are trained with the three data sets described above to either minimize the regular GAN objective described in section II-A or to minimize the EM distance with gradient penalty described in section II-E.

C. Extra Domain Classification

The experiments described here are performed to study whether data of $\mathcal{A}^c$ can be classified accurately when encountered during testing. To do so, firstly, CoAC-GANs and AC-GANs trained as described in section IV-B are used to generate data from all possible domain-class combinations, including the ones missing from the training set. This fake data is used to construct two types of training sets. One consists of only fake data and one consists of both fake and real data. For both of these types of data sets, for both the regular GAN and WGAN objectives, five CoAC-GANs and five AC-GANs were trained with and tested on their ability to generate the missing data from MNIST^1 and MNIST^2. This results in a total of $2 \times 2 \times 2 \times 2 \times 5 = 80$ generative models. For each of these generative models, an individual classifier is trained.

a) Class label distributions: The class label distribution of a data set consisting of only fake data used for classifier training is taken from the matching real dataset. It is assumed that the frequency in which the missing domain-class instances occur is either known or can be estimated well. Missing domain-label frequencies thus follow the corresponding real data sets as well.

For a data set that contains both real and fake data the distribution of class labels for the real data is simply that of domain $\mathcal{A}$. The distribution of class labels for the fake data only differs from that of the data set containing only fake data in that the amount of data with labels that only occur in $\mathcal{B}$ is doubled. This is done to mimic the expected class label distribution during testing, assuming that the missing classes in domain $\mathcal{A}$ will occur proportionally to those of domain $\mathcal{B}$.

b) Baselines: The accuracy of naively trained classifiers on class-domain combinations that were not presented during training is also examined. These classifiers are trained on only data from domain $\mathcal{B}$ and on the junction of $\mathcal{A}$ and $\mathcal{B}$, where again labels of classes that are represented in both $\mathcal{A}$ and $\mathcal{B}$ are sampled with equal probability during training. Five classifiers were trained for each of these types of training sets for both MNIST^1 and MNIST^2. The performance of the CoAC-GANs and AC-GANs are compared with these baselines.

TABLE I
GENERATOR FOR GANS TRAINED WITH CELEBA IMAGES

Layer	Input	Type	Output	Description
0	-	Input	$100 + d$	-
1	0	FC	$512 \times 4 \times 4$	BN, ReLU
2	1	tconv	$256 \times 8 \times 8$	k5, s2, p2, BN, ReLU
3	2	tconv	$128 \times 16 \times 16$	k5, s2, p2, BN, ReLU
4	3	tconv	$64 \times 32 \times 32$	k5, s2, p2, BN, ReLU
5	4	tconv	$3 \times 64 \times 64$	k5, s2, p2, tanh

D. Model details

1) GANs:

a) Architectures: The architectures of the GANs trained with CelebA images in this work follow the architectural constraints of DCGAN [16]. Tables I and II show their details. Tables III and IV show the architectures details of GANs trained with digit images. These architectures were also used in [3]. In Tables I, II, III, and IV the 'Input' column denotes the preceding layer and the 'Output' column denotes the output dimensions of a layer. Furthermore, abbreviations have been used for fully connected layer (FC), convolution layer [10], [17] (conv), transposed convolution layer [18], [19] (tconv), max pooling layer [10] (pool), batch normalization [20] (BN), rectified linear unit [21] (ReLU), leaky ReLU [22] with a slope of 0.2 (LReLU), and parametrized ReLU [23] (PReLU). The kernel size x, stride y and amount of padding z are denoted by kx, sy, and pz respectively.

Class labels are represented as one-hot vectors. For Tables II and IV layer 6 and 7 respectively is only added for AC-GANs or CoAC-GANs. When predicting multiple class labels, multiple instances of these layers are present in the architectures. For each of these labels, c denotes the length of the corresponding one-hot vector. d in Tables I and III denotes the length of the concatenation of all one-hot vectors that are presented to a generator.

For a CoGAN or CoAC-GAN, layers 1-4, 2-6, 1-4 and 2-7 are shared between all models for the architectures in Tables I, II, III and IV respectively.

In all experiments, instance normalization [24] replaces batch normalization in the WGAN critics. Following [14], the sigmoid activation in the last layer of all WGAN discriminators is omitted.

b) Training: The GANs are trained with the Adam optimization algorithm [25] with a learning rate of 0.0002 and decay rates for the first- and second-order moment estimates $\beta_1 = 0.5$, and $\beta_2 = 0.999$. The batch size is set to 64 samples. For a CoGAN or CoAC-GAN, a single update consists of a forward pass and backward pass through each individual GAN that the model is composed of, which results in an effective batch size of 128 samples for their shared layers. All

TABLE II
DISCRIMINATOR FOR GANS TRAINED WITH CELEBA IMAGES

Layer	Input	Type	Output	Description
0	-	Input	$3 \times 64 \times 64$	-
1	0	conv	$64 \times 32 \times 32$	k5, s2, p2, LReLU
2	1	conv	$128 \times 16 \times 16$	k5, s2, p2, BN, LReLU
3	2	conv	$256 \times 8 \times 8$	k5, s2, p2, BN, LReLU
4	3	conv	$512 \times 4 \times 4$	k5, s2, p2, BN, LReLU
5	4	conv	$1 \times 1 \times 1$	k4, s1, p0 sigmoid
6	4	conv	$c \times 1 \times 1$	kc, s1, p0, softmax

TABLE III
GENERATOR FOR GANS TRAINED WITH DIGIT IMAGES

Layer	Input	Type	Output	Description
0	-	Input	$(100 + d) \times 1 \times 1$	-
1	0	tconv	$512 \times 4 \times 4$	k4, s1, p0, BN, PReLU
2	1	tconv	$256 \times 7 \times 7$	k3, s2, p1, BN, PReLU
3	2	tconv	$128 \times 13 \times 13$	k3, s2, p1, BN, ReLU
4	3	tconv	$64 \times 25 \times 25$	k3, s2, p1, BN, ReLU
5	4	tconv	$1 \times 28 \times 28$	k4, s1, p1, tanh

AC-GANs are trained for 100 epochs. CoGANs and CoAC-GANs are trained for 125,000 batches on digit data sets and for 150,000 batches on CelebA. For each five batches with a discriminator update, one generator update is done.

2) Classifiers: The architecture of the models trained for classification is a variant of the LeNet [26] architecture. It is shown in Table V. The classifiers are trained with SGD with a learning rate of 0.01 with 64 samples per batch. To keep the classifiers from overfitting, 10,000 samples are removed from the training set and used as a validation set. During training, the error on the validation set is monitored at each epoch. When the validation error is not improved upon for 10 epochs, training is terminated and the model at the epoch with the lowest validation error is used for testing.

TABLE IV
DISCRIMINATOR FOR GANS TRAINED WITH DIGIT IMAGES

Layer	Input	Type	Output	Description
0	-	Input	$1 \times 28 \times 28$	-
1	0	conv	$20 \times 24 \times 24$	k5, s1, p0
2	1	pool	$20 \times 12 \times 12$	k2, s1, p0
3	2	conv	$50 \times 8 \times 8$	k5, s1, p0
4	3	pool	$50 \times 4 \times 4$	k2, s1, p0
5	4	conv	$100 \times 1 \times 1$	k4, PreLU
6	5	FC	1	sigmoid
7	5	FC	c	softmax

TABLE V
CLASSIFIER

Layer	Input	Type	Output	Description
0	-	input	$1 \times 28 \times 28$	-
1	0	conv	$20 \times 24 \times 24$	k5, s1, p0
2	1	pool	$20 \times 12 \times 12$	k2, s1, p0, ReLU
3	2	conv	$50 \times 8 \times 8$	k5, s1, p0
4	3	pool	$50 \times 4 \times 4$	k2, s1, p0, ReLU
5	4	FC	500	ReLU
6	5	FC	c	softmax

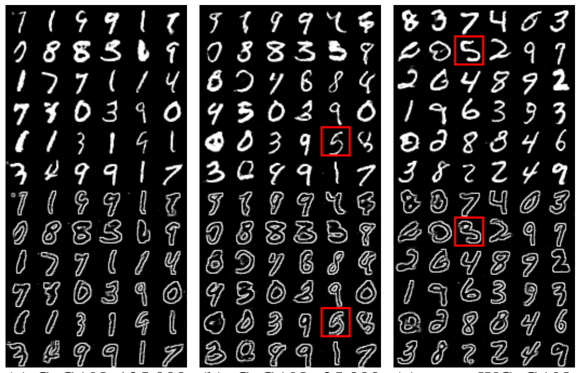

(a) CoGAN, 125,000 batches. (b) CoGAN, 25,000 batches. (c) WCoGAN, 125,000 batches.

Fig. 1. Images generated by CoGANs trained on MNIST[1]. The top images show digits produced by the GAN trained on MNIST data. The images below them show the corresponding digits produced by the GAN trained on MNIST-edge data. These digits were produced from the same input noise vectors. The noise vectors used to produce the digits in figures 1a and 1b are also identical. Strong cases of the types of images discussed in section V-A1 are indicated with a red border.

V. RESULTS

In this section, we denote a GAN that is trained with the WGAN objective with the prefix 'W'. When this prefix is absent, the regarding model is trained with the regular GAN objective.

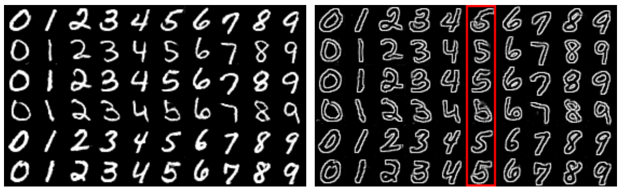
A. Extra Domain Data Generation

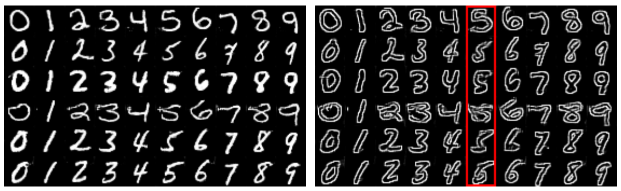
1) Digits: Figure 1 shows images produced by a CoGAN and WCoGAN trained on MNIST[1]. At the end of training, the CoGAN has collapsed to produce a disproportionally large share of data from a single mode, which is observed from the large number of digits '1' in figure 1a. At the optimal point during the training, the CoGAN produces images that cover a larger variety of digits. Interestingly, for some input noise vectors the CoGAN is also able to produce the digit '5' in both domains, even though this was missing in the training data of the MNIST-edge domain.

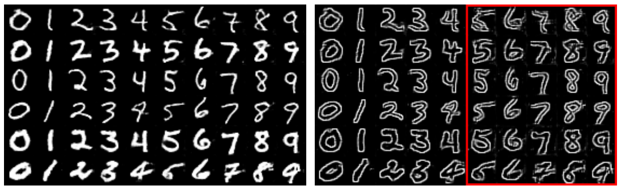
Training a WCoGAN does not result in a mode collapse. Also, after training is completed, the model is able to produce the digit '5' in the MNIST domain. However, the corresponding images in the MINST-edge domain do not resemble '5's. Instead, the model produces loosely coupled images that resemble other digits.

Fig. 2 shows images produced by some of the other architectures trained on MNIST[1] and MNIST[2]. We found that for MNIST[1] all models were able to generate recognizable instance of the digit '5' of MNIST-edge that was missing in this training set. For MNIST[2], only CoAC-GANs and WCoAC-GANs were able to consistently produce recognizable instances of the digits '5'-'9' from MNIST-edge.

2) Faces: We found that the CoGAN and WCoGAN trained with CelebA[1] were able to generate images of smiling and non-smiling females and non-smiling males, but were unable to generate images of smiling males, which were missing in the training data. Fig. 3 shows images produced by AC-GANs


(a) MNIST[1], CoAC-GAN, 125,000 batches.


(b) MNIST[1], WAC-GAN, 100 epochs.


(c) MNIST[2], WCoAC-GAN, 125,000 batches.

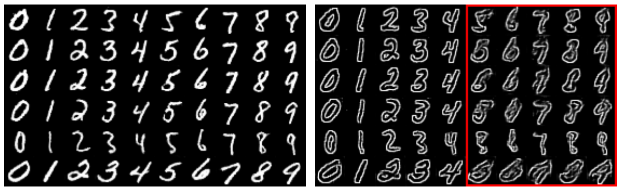

(d) MNIST[2], AC-GAN, 100 epochs.

Fig. 2. Images generated by different GAN architectures trained on MNIST[1] and MNIST[2]. The digits on the left and right half of the image were produced from the same input noise vectors. Attempts at generating images of missing domain-class combinations are indicated with a red border.

and CoAC-GANs. CoAC-GANs were also unable to generate images of the missing domain-class combination. The quality of the generated images is inferior to those generated by a CoGAN in the same setting. CoAC-GAN is also prone to mode collapse, which clearly shows in all smiling mouths of the images it produced. The images of smiling males generated early in the training process in Fig. 3b show color artifacts in the mouth area that are not represented in the training data. Furthermore, the difference between non-smiling males and females that is present in early stages of training almost completely disappears at the end of the training process.

The trained WCoAC-GANs are able to produce images of smiling males, especially in the early stages of training as can be seen in Fig. 3f. In Fig. 3h color artifacts can again be seen around the mouths of the images of smiling males produced at the end of training. As with CoAC-GANs, these images also show more female characteristics than those obtained earlier in the training process.

The results produced by AC-GANs and WAC-GANs show none of the issues of CoAC-GANs described above. These models are able to produce images of smiling males even though they have not been presented with this domain-class combination during training.

B. Extra Domain Classification

Tables VI and VII show the accuracy of the classifiers trained on only real data, on only fake data and on hybrid data sets. Table VI shows the accuracy on ‘5’s from MNIST-edge. These results were obtained from classifiers trained on real data that originates from MNIST[1] and/or fake training data produced by GANs trained on MNIST[1]. Table VII shows the accuracy on the digits ‘5’-‘9’ from MNIST-edge. These results were obtained from classifiers trained on real data that originates from MNIST[2] and/or fake training data produced by GANs trained on MNIST[2]. The best performing classifiers are printed in bold. Both tables show that the baseline classifiers trained on only real data clearly classify data from the missing class-domain combinations more accurately than a random assignment of labels based on prior class probabilities.

Training on data produced by CoAC-GANs rather than on that produced by AC-GANs results in a better classifier performance in all experiments. In general, training only on fake data results in a better accuracy than training on hybrid data with both real and fake samples.

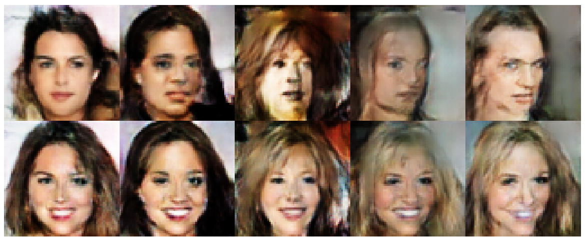
For classifying missing images from MNIST[1], the performance of the baseline is surpassed by all classifiers trained on fake or hybrid datasets. For classifiers tasked to classify missing digits from MNIST[2], this only holds when the fake training data originates from CoAC-GANs. Classifiers tasked to classify missing digits from MNIST[2] trained on fake data originating from AC-GANs have poor accuracy. The fact that a better accuracy is obtained by classifiers trained on data generated by CoAC-GAN with respect to those trained on data generated by AC-GANs coincides with the quality of the images shown in Fig. 2.

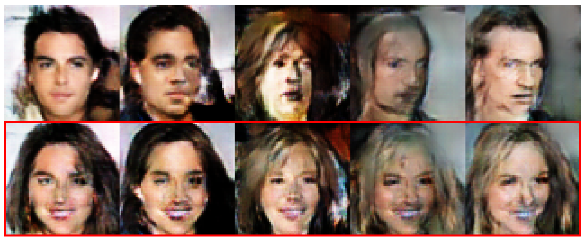
TABLE VI
CLASSIFIER ACCURACY ON THE MISSING DIGITS ‘5’ OF MNIST-EDGE FROM MNIST[1]

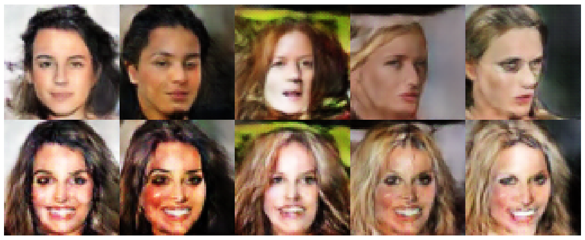
Train data	Objective	Type	Accuracy (%)
real MNIST	-	-	40.45 ± 7.42
real MNIST + real MNIST-edge	-	-	53.34 ± 6.82
fake MNIST-edge	WGAN	AC	94.69 ± 4.22
		CoAC	97.00 ± 1.05
	GAN	AC	97.71 ± 0.40
		CoAC	$\mathbf{98.03 \pm 0.70}$
real MNIST-edge + fake MNIST-edge	WGAN	AC	89.75 ± 6.44
		CoAC	96.82 ± 0.82
	GAN	AC	94.13 ± 1.58
		CoAC	97.13 ± 0.92

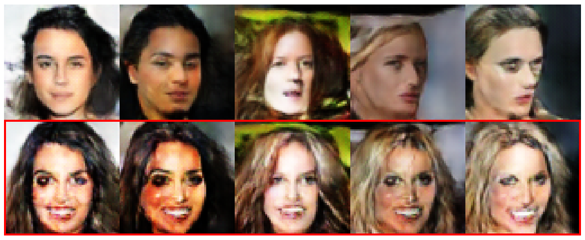
VI. DISCUSSION

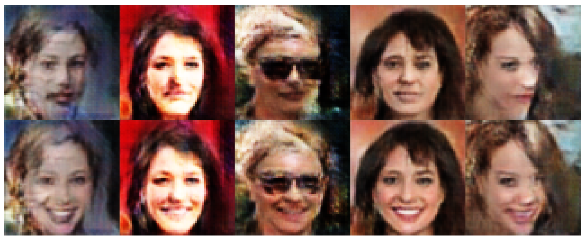
Images generated by CoAC-GANs of the domain-class combination that are missing in CelebA[1] show undesirable artifacts in the area that characterize the regarding class. This might be attributed to the CoGAN architecture. The domain specific information in the generators is forced to reside in the non-shared weights of the CoGAN, but this is not the case for

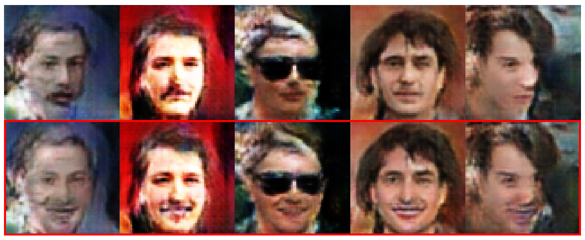

(a) CoAC-GAN, female, 25,000 batches.

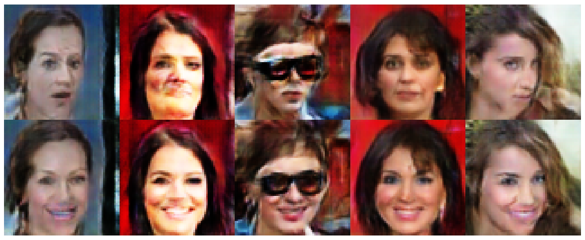

(b) CoAC-GAN, male, 25,000 batches.

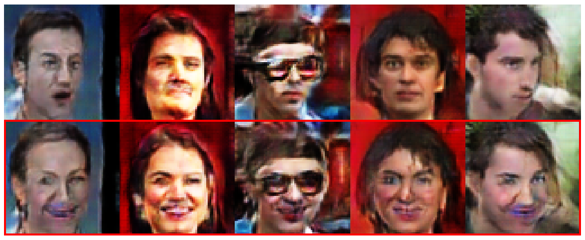

(c) CoAC-GAN, female, 150,000 batches.

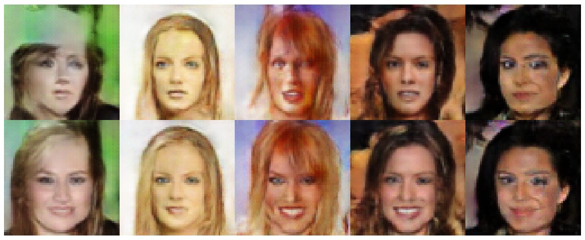

(d) CoAC-GAN, male, 150,000 batches.

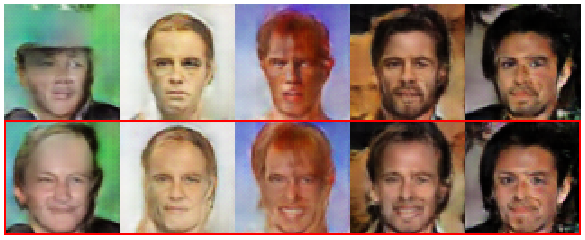

(e) WCoAC-GAN, female, 25,000 batches.

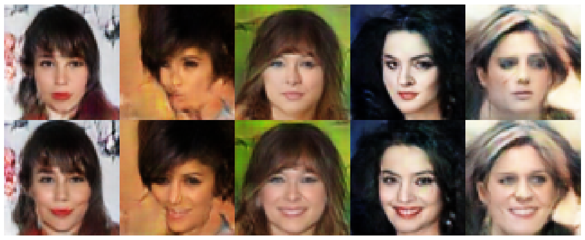

(f) WCoAC-GAN, male, 25,000 batches.


(g) WCoAC-GAN, female, 150,000 batches.


(h) WCoAC-GAN, male, 150,000 batches.


(i) AC-GAN, female, 100 epochs.


(j) AC-GAN, male, 100 epochs.


(k) WAC-GAN, female, 100 epochs.

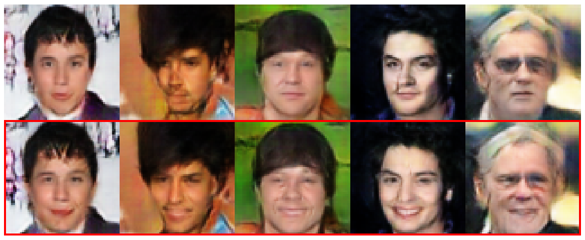

(l) WAC-GAN, male, 100 epochs.

Fig. 3. Images generated by GANs trained on CelebA[1]. The bottom rows contain images generated by GANs when they were primed to generate smiling faces and top rows contain images generated by GANs when they were primed to generate non-smiling faces. Attempts at generating images of the missing domain-class combination are indicated with a red border.

TABLE VII
CLASSIFIER ACCURACY ON THE MISSING DIGITS '5'-'9' OF MNIST-EDGE FROM MNIST^2

Train data	Objective	Type	Accuracy (%)
real MNIST	-	-	67.74 ± 24.69
real MNIST + real MNIST-edge	-	-	45.14 ± 14.11
fake MNIST-edge	WGAN	AC	68.04 ± 14.50
		CoAC	$\mathbf{95.82 \pm 1.60}$
	GAN	AC	56.26 ± 29.80
		CoAC	93.27 ± 3.80
real MNIST-edge + fake MNIST-edge	WGAN	AC	57.14 ± 18.47
		CoAC	94.33 ± 1.77
	GAN	AC	39.16 ± 27.16
		CoAC	92.45 ± 4.12

class specific information. One explanation could therefore be that the decoding of information specific to the class label that is missing in domain $\mathcal{A}$ happens partly in the shared and partly in the non-shared generator weights. This is especially likely when the class label is not solely characterized by low level image details. However, the non-shared part of the generator that is tasked to generate images from domain $\mathcal{A}$ is never trained to further encode output of the shared layers when these are presented with the class label that is missing in domain $\mathcal{A}$. It would in this case be unable to transform this shared layer output into meaningful images.

VII. CONCLUSION

This paper shows that it is possible to use AC-GANs and CoAC-GANs trained with the regular GAN or WGAN objective to generate new data that seems to originate from some domain, but characterizes classes of which no data is available in that domain. In order to do so, data characterizing these classes must be available in some other similar domain.

Despite the theoretical downsides of using CoGANs for this task (see section III-A), in some cases, CoGANs are able to generate data from the missing classes. However, CoGANs cannot be presented with conditional class variables and therefore it cannot generate labeled output.

The preference for AC-GANs or CoAC-GANs depends on the training data. CoAC-GANs seem to have a superior performance when the domains only differ in low level detail and for such domains seem to be able to cope better with multiple missing classes than AC-GANs. This is in line with theoretical expectations due to the difference in shared weights of the GANs that make up the architectures of CoAC-GANs or AC-GANs.

We have also seen that in some cases, the generated data can be used to train classifiers that accurately classify the missing classes when they are encountered after training.

REFERENCES

[1] Kingma, Diederik P and Welling, Max, "Auto-Encoding Variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[2] Goodfellow, Ian and Pouget-Abadie, Jean and Mirza, Mehdi and Xu, Bing and Warde-Farley, David and Ozair, Sherjil and Courville, Aaron and Bengio, Yoshua, "Generative Adversarial Nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[3] Liu, Ming-Yu and Tuzel, Oncel, "Coupled Generative Adversarial Networks," in *Advances in neural information processing systems*, 2016, pp. 469–477.

[4] Odena, Augustus and Olah, Christopher and Shlens, Jonathon, "Conditional Image Synthesis With Auxiliary Classifier GANs," *arXiv preprint arXiv:1610.09585*, 2016.

[5] Isola, Phillip and Zhu, Jun-Yan and Zhou, Tinghui and Efros, Alexei A, "Image-to-Image Translation With Conditional Adversarial Networks," *arXiv preprint*, 2017.

[6] Choi, Yunjey and Choi, Minje and Kim, Munyoung and Ha, Jung-Woo and Kim, Sunghun and Choo, Jaegul, "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation," *arXiv preprint arXiv:1711.09020*, 2017.

[7] Ziwei Liu and Ping Luo and Xiaogang Wang and Xiaoou Tang, "Deep Learning Face Attributes in the Wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[8] LeCun, Yann, "The MNIST database of handwritten digits," *http://yann.lecun.com/exdb/mnist/*, 1998.

[9] Rumelhart, David E and Hinton, Geoffrey E and Williams, Ronald J, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[10] LeCun, Yann and Boser, Bernhard E and Denker, John S and Henderson, Donnie and Howard, Richard E and Hubbard, Wayne E and Jackel, Lawrence D, "Handwritten Digit Recognition with a Back-Propagation Network," in *Advances in neural information processing systems*, 1990, pp. 396–404.

[11] Sabour, Sara and Frosst, Nicholas and Hinton, Geoffrey E, "Dynamic Routing Between Capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3859–3869.

[12] Pieters, Mathijs and Wiering, Marco, "Comparing Generative Adversarial Network Techniques for Image Creation and Modification," *arXiv preprint arXiv:1803.09093*, 2018.

[13] Upadhyay, Yash and Schrater, Paul, "Generative Adversarial Network Architectures For Image Synthesis Using Capsule Networks," *arXiv preprint arXiv:1806.03796*, 2018.

[14] Arjovsky, Martin and Chintala, Soumith and Bottou, Léon, "Wasserstein GAN," *arXiv preprint arXiv:1701.07875*, 2017.

[15] Gulrajani, Ishaan and Ahmed, Faruk and Arjovsky, Martin and Dumoulin, Vincent and Courville, Aaron C, "Improved Training of Wasserstein GANs," in *Advances in Neural Information Processing Systems*, 2017, pp. 5769–5779.

[16] Radford, Alec and Metz, Luke and Chintala, Soumith, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *arXiv preprint arXiv:1511.06434*, 2015.

[17] Goodfellow, Ian and Bengio, Yoshua and Courville, Aaron and Bengio, Yoshua, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[18] Long, Jonathan and Shelhamer, Evan and Darrell, Trevor, "Fully Convolutional Networks For Semantic Segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[19] Dumoulin, Vincent and Visin, Francesco, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[20] Ioffe, Sergey and Szegedy, Christian, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv preprint arXiv:1502.03167*, 2015.

[21] Nair, Vinod and Hinton, Geoffrey E, "Rectified Linear Units Improve Restricted Boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[22] Maas, Andrew L and Hannun, Awni Y and Ng, Andrew Y, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

[23] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian, "Delving deep into rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[24] Dmitry Ulyanov and Andrea Vedaldi and Victor S. Lempitsky, "Instance Normalization: The Missing Ingredient for Fast Stylization," *CoRR*, vol. abs/1607.08022, 2016.

[25] Kingma, Diederik P and Ba, Jimmy, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] LeCun, Yann and Bottou, Léon and Bengio, Yoshua and Haffner, Patrick, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.