Computer Vision and Image Understanding 186 (2019) 1–12

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Generalizing semi-supervised generative adversarial networks to regression using feature contrasting☆

Greg Olmschenk [a,b,*], Zhigang Zhu [a,b], Hao Tang [c]

[a] The City College, The City University of New York, 160 Convent Ave, New York, NY 10031, USA
[b] The Graduate Center, The City University of New York, 365 5th Ave, New York, NY 10016, USA
[c] Borough of Manhattan Community College, The City University of New York, 199 Chambers St, New York, NY 10007, USA

ARTICLE INFO

Keywords:
Generative adversarial learning
Age estimation
Regression

ABSTRACT

In this work, we generalize semi-supervised generative adversarial networks (GANs) from classification problems to regression problems. In the last few years, the importance of improving the training of neural networks using semi-supervised training has been demonstrated for classification problems. We present a novel loss function, called feature contrasting, resulting in a discriminator which can distinguish between fake and real data based on feature statistics. This method avoids potential biases and limitations of alternative approaches. The generalization of semi-supervised GANs to the regime of regression problems of opens their use to countless applications as well as providing an avenue for a deeper understanding of how GANs function. We first demonstrate the capabilities of semi-supervised regression GANs on a toy dataset which allows for a detailed understanding of how they operate in various circumstances. This toy dataset is used to provide a theoretical basis of the semi-supervised regression GAN. We then apply the semi-supervised regression GANs to a number of real-world computer vision applications: age estimation, driving steering angle prediction, and crowd counting from single images. We perform extensive tests of what accuracy can be achieved with significantly reduced annotated data. Through the combination of the theoretical example and real-world scenarios, we demonstrate how semi-supervised GANs can be generalized to regression problems.

1. Introduction

Deep learning (LeCun et al., 2015), particularly deep neural networks (DNNs), has become the dominant focus in many areas of computer science in recent years. This is especially true in computer vision, where the advent of convolutional neural networks (CNNs) (LeCun et al., 1999) has led to algorithms which can outperform humans in many vision tasks (Dodge and Karam, 2017). Within the field of deep learning, generative models have become popular for generating data that simulates real datasets. A generative model is one which learns how to produce samples from a data distribution. In the case of computer vision, this is often a neural network which learns how to generate images, possibly with specified characteristics. Generative models are particularly interesting because for such a model to generate new examples of data from a distribution, the model must be able to distinguish data which belongs to the distribution and that which does not. In a sense, this distinguishing ability shows that the network "understands" a data distribution. Arguably the most powerful type of generative model is the generative adversarial network (GAN) (Goodfellow, 2016; Goodfellow et al., 2014). GANs have been shown to be capable of producing fake data that appears to be real to human evaluators. For example, GANs can generate fake images of real world objects which a human evaluator cannot distinguish from true images (Elsayed et al., 2018). Beyond this, GANs have been shown to produce better results in discriminative tasks using relatively small amounts of data (Salimans et al., 2016), where equivalent DNNs/CNNs would require significantly more training data to accomplish the same level of accuracy. As one of the greatest obstacles in deep learning is acquiring the large amount of labeled data to train such models, the ability to train these powerful models with much less data is of immense importance.

While GANs have already shown significant potential in semi-supervised training, they have only been used for a limited number of cases. In particular, they have almost exclusively been used for classification problems thus far. In this work, we propose generalizing semi-supervised GANs to regression problems. Though this may initially seem to be a trivial expansion, the nature of a GAN's optimization

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.cviu.2019.06.004.

* Corresponding author at: The Graduate Center, The City University of New York, 365 5th Ave, New York, NY 10016, USA.
E-mail address: golmschenk@gradcenter.cuny.edu (G. Olmschenk).

https://doi.org/10.1016/j.cviu.2019.06.004
Received 1 July 2018; Received in revised form 15 February 2019; Accepted 27 June 2019
Available online 5 July 2019

goals makes the shift from classification to regression problems difficult. Specifically, the two parts of a GAN can be seen as playing a minimax game. The discriminating portion of the GAN must have the objective of labeling the fake data from generating portion as fake. In a classification semi-supervise GAN, an additional "fake" class is added to the possible list of classes. However, in regression, when the data is labeled with real valued numbers, deciding on what constitutes a "fake" labeling is not straight forward.

1.1. Contributions

In this work, we will present the following contributions:

1. A new algorithm with a novel loss function, feature contrasting, which allows semi-supervised GANs to be applied to regression problems, the Semi-supervised Regression GAN (SR-GAN).
2. A set of optimization rules which allows for stable, consistent training when using the SR-GAN, including experiments demonstrating the importance of these rules.
3. Systematic experiments using the SR-GAN on the real world applications of age estimation, driving steering angle prediction, and crowd counting from single images showing the benefits of SR-GANs over existing approaches.

The most important contribution is the introduction of the generalized semi-supervised regression GAN (SR-GAN) formulation using feature contrasting. Nevertheless, while the theoretical solution for applying semi-supervised GANs to regression is provided in the first contribution, there are several factors that need to be addressed for this approach to work in practice. Chiefly is the stability of training the two competing networks in an SR-GAN. This is addressed by designing loss functions for the SR-GAN whose gradients are well-behaved (neither vanishing nor exploding) in as many situations as possible, and preventing cyclical training between the generator and discriminator by applying penalties and limitations in the training behavior.

We provide a number of real world applications where SR-GANs are shown to improve the results over traditional CNNs and other competing models. Specifically we will use the SR-GAN to predict the age of an individual, estimate the angle a steering wheel should be turned to given an image of the upcoming road segment, and count the size of a crowd from a single image. The age estimation and steering angle datasets provides relatively simple applications on which the SR-GAN can be used to reduce the data requirements in a real world situation, while still being challenging and general enough to merit attention. The crowd counting application provides a more complex scenario with a wide variety of conditions to show the method in more difficult circumstances.

1.2. Outline

The remainder of the paper is laid out as follows. The work which our method builds off of as a starting point and other related works are examined in Section 2. Section 3 explains our methods and experimental setup. Section 4 displays the experimental results and discusses the findings. Finally, we conclude in Section 5.

2. Background and related work

2.1. The value of regression problems

Regression problems encompass a large pool of applications which cannot be solved – or would be poorly solved – by framing them as classification problems. The SR-GAN as we define it here can be generalized to any such regression problem. Some examples include crowd counting estimation (Zhang et al., 2015), weather prediction models (Xingjian et al., 2015), stock index evaluation (Ding et al., 2015), object distance estimation (Eigen et al., 2014), age estimation (Niu et al., 2016),

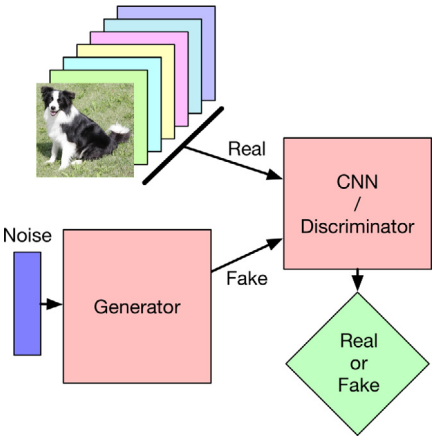


Fig. 1. The structure of a basic GAN. Real and fake images are fed to a discriminator network, which tries to determine whether the images are real or fake. The fake images are produced by a generator network.

data hole filling (Pathak et al., 2016), curve coefficient estimation, ecological biomass prediction (Ali et al., 2015), traffic flow density prediction (Lv et al., 2015), orbital mechanics predictions (Hartikainen et al., 2012), electrical grid load prediction (Marino et al., 2016), stellar spectral analysis (Fabbro et al., 2017), network data load prediction (Oliveira et al., 2016), object orientation estimation (Schwarz et al., 2015), species population prediction (Bland et al., 2015), ocean current prediction (Liu and Weisberg, 2005), and countless others. While it is possible to frame each of these problems in terms of classification, in practice, this presents several significant problems. For example, the developer must decide on an arbitrary number of classes for the application. However, more importantly, such a naive classification approach results in each incorrect prediction being considered equally as erroneous. In regression applications, the true label lies somewhere on a continuous scale, and the closer of two predictions should always be considered better than the farther, even if both are inaccurate. If the prediction of a real number from 0 to 10 was split into 10 discrete classes, a prediction of 8 should be considered better than a prediction of 2 for a true label of 10. Yet, a naive classification network produces the same loss for each. Depending on the accuracy required by the application, this approach may be acceptable, but these problems are more naturally framed as regression problems.

2.2. Generative adversarial networks

A Generative Adversarial Network (GAN) consists of two neural networks which compete against one another. One of the networks generates fake data; hence we will call it the generator. The other network attempts to distinguish between real data and the fake generated data; consequently, this network is called the discriminator. Both networks are trained together, each continually working to outperform the other and adapting in accordance to the other.

Though GANs are now fairly common, to provide the groundwork for our SR-GAN, it is worth defining the details of a GAN from the viewpoint of probability distributions. Although these methods work for any prediction application, to give a concrete understanding, these explanations are given in terms of computer vision problems, specifically where the datasets consist of images. This means an example of real data (and thus the input of the discriminator) is an image, and the output of the generator is also an image. The structure of a GAN can be seen in Fig. 1.

The generator network takes random noise as input (usually sampled from a normal distribution) and outputs the fake image data. The discriminator takes as input images and outputs a binary classification of either fake or real data. Images can be represented by a vector, with

each element representing the value of a pixel in the image.[1] In any image, each element of this vector has a value within a certain range representing the intensity of that pixel. For this explanation, we will state the minimum element value (pixel value) as being 0, and the maximum as being 1. Of course, this vector can be represented as a point in N dimensional space, where N is the number of elements in the vector. The possible positions of an image's point are restricted to the N dimensional hypercube with a side length of 1. Here, it is important to note that real-world images are not equally spread throughout this cube. That is, most points in the cube correspond to images that would look like random noise to a human. Images from the real world usually have properties like local consistency in both texture and color, logical relative positioning of shapes, etc. Real world images lie on a manifold within the cube (Fefferman et al., 2016). Subsets of real-world images, such as the set of all images containing a dog, lie on yet a smaller manifold. This manifold represents a probability distribution of the real world images. We can view the real world as a data generating probability distribution, with each position on the manifold having a certain probability based on how likely that image is to exist in the real world.

The goal of the generator is then to produce images which match the probability distribution of the manifold as closely as possible. Input to the generator is a point sampled from the probability distribution of (multidimensional) random normal noise, and the output is a point in the hypercube—an image. The generator is then a function which transforms a normal distribution into an image data distribution. Formally,

$$p_{fake}(\boldsymbol{x}) = G(\mathcal{N}) \tag{1}$$

where G represents the generator function, $\boldsymbol{x}$ is a random variable representing an image, $\mathcal{N}$ is the normal distribution, and $p_G(\boldsymbol{x})$ is the probability distribution of the images generated by the generator. The desired goal of the generator is to minimize the difference between the generated distribution and the true data distribution. One of the most common metrics to minimize this difference is the Kullback–Leibler (KL) divergence between the generator distribution and the true data distribution using maximum likelihood estimation. This is done by finding the parameters of the generator, θ, which produce the smallest divergence,

$$\theta^* = \arg\min_{\theta} D_{\mathrm{KL}}(p_{data}(\boldsymbol{x}) \parallel p_G(\boldsymbol{x};\theta)). \tag{2}$$

To find this set of parameters, each of the discriminator and the generator works toward minimizing a loss function. For the discriminator, the loss function is given by

$$L_D = -\mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p_{fake}(\boldsymbol{x})}[\log(1 - D(\boldsymbol{x}))] \tag{3}$$

and the generator's loss function is given by

$$L_{fake} = -\mathbb{E}_{\boldsymbol{x}\sim p_{fake}(\boldsymbol{x})}[\log(D(\boldsymbol{x}))]. \tag{4}$$

In the case of image data, this approach has led to generative models which can produce realistic looking images reliably (Radford et al., 2015).

2.3. Semi-supervised GANs for classification

In this section, we will explain a subset of GANs which are used to improve the training of ordinary networks for discrimination and prediction tasks. In this case, both a labeled and an unlabeled dataset is used, and in addition to distinguishing between real and fake, the discriminator also tries to label a real input data sample into one of the given classes. The primary goal of this type of GAN is to allow the discriminator's prediction task to be trained with relatively small amounts of labeled data using unlabeled data to provide the network with additional information. As unlabeled data is usually much easier to obtain than labeled data, this provides a powerful means to reduce the requirements of training neural networks. This semi-supervised GAN structure can be seen in Fig. 2.

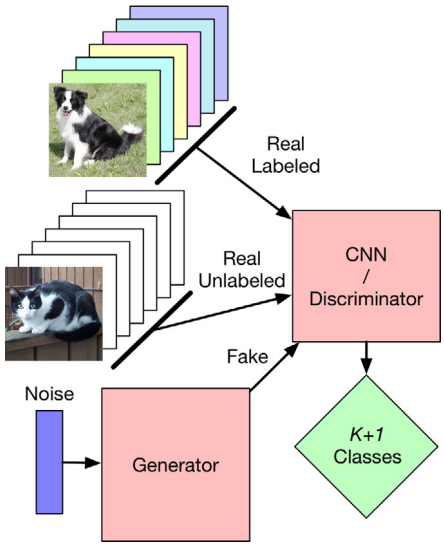


Fig. 2. The structure of a semi-supervised GAN. Both labeled and unlabeled real images, as well as fake images, are fed to a discriminator network, which tries to determine which class each image belongs to (K real classes and one fake class). The discriminator wishes to label images from the generator as belonging to a special "fake" class.

Where in a simple GAN the discriminator would be passed true examples and fake examples, in the semi-supervised GAN the discriminator is given true labeled examples, true unlabeled examples, and fake examples. We can better understand why this is useful by considering the case of image classification. In this case, the discriminator is being trained to predict the correct class of a true image, which can be one of the K classes that exist in the dataset. The discriminator is given the additional goal of attempting to label any fake images with a $K + 1$th class, which only exists to label fake data (i.e., does not exist in the true label dataset). For the case of unlabeled, all we know is that it must belong to one of the first K classes, as the $K + 1$th class does not exist in the real data. The discriminator is then punished for labeling true unlabeled data as the $K + 1$th class. This is useful because the discriminator cannot simply overfit to the labeled data, as it still has to accommodate for the unlabeled data. At the same time, the fake data prevents the discriminator from allowing simple features to be the deciding factor, as the generator is able to produce such simple features.

To understand what is happening in this semi-supervised learning more intuitively, we can imagine the extreme case of an ideal discriminator and generator. The generator would have to have learned to produce data which exactly matches the true data distribution. For this to happen, the discriminator must have forced the generator to learn this (as the generator's training is entirely dictated by backpropagation from the discriminator), meaning the discriminator too "knows" exactly the data distribution. If there were any difference between the true and generated image distributions, the discriminator could use this to distinguish between real and fake, and then the generator could still be trained further toward producing a match of the true distribution.

Viewing this from the perspective of the manifold in data space again, there are few labeled data points and many unlabeled data points which must lie on the manifold. The manifold has different regions (or even separate manifolds) for each class, but even the unlabeled data has to lie somewhere on the manifold. As the discriminator trains, it learns

[1] One element per pixel is in the case of grayscale images. For RGB images, there will be three elements in the vector for each pixel, one for each color channel of the pixel.

how to segment the data points into categories. To do this, it creates a mapping from a predictive manifold to a class, with the training warping the manifold to contain each of the data points for that class. At the same time, the generator prevents the manifold from warping too severely to reach data points in arbitrary ways. Intuitively, this is because severely warping the manifold to reach true data points can result in the manifold stretching into the area which does not represent true images. The generator acts a pressure on the manifold to reduce this. By generating images near the manifold, the generator forces the discriminator's manifold not to wander into areas that do not contain real images. In this sense, the generator is a form of regularization for the discriminator, but one which is based on real-world data.

As originally formulated by Salimans et al. (2016), the discriminator loss function is then defined by

$$L_D = L_{supervised} + L_{unsupervised} \tag{5}$$

$$L_{supervised} = - \mathbb{E}_{\boldsymbol{x},y\sim p_{labeled}(\boldsymbol{x},y)} \log[p_{model}(y \mid \boldsymbol{x}, y < K+1)] \tag{6}$$

$$L_{unsupervised} = - \mathbb{E}_{\boldsymbol{x}\sim p_{unlabeled}(\boldsymbol{x})} \log[1 - p_{model}(y = K+1 \mid \boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x}\sim p_{fake}} \log[p_{model}(y = K+1 \mid \boldsymbol{x})]. \tag{7}$$

As for the generator, the first option for a loss function is the straight forward one which aims to have the discriminator label the fake images as from real classes. Specifically,

$$L_G = -\mathbb{E}_{\boldsymbol{x}\sim p_{fake}} \log[p_{model}(y < K+1 \mid \boldsymbol{x})]. \tag{8}$$

However, Salimans et al. (2016) found better results by trying to have the output activations of an intermediate layer of the discriminator have similar statistics in both the fake and real image cases. That is, the generator should try to make its images produce similar features in an intermediate layer as is produced when true images are input. This can be intuitively understood as making the statistics of the image be the same in both the fake and real cases, specifically, the feature statistics that are used in deciding a classification. The simplest and most useful statistic to try to match is the expected value for each feature. Formally put, if we denote $f(\boldsymbol{x})$ as the features output by an intermediate layer in the discriminator, then the loss function for the generator becomes

$$L_G = \left\| \mathbb{E}_{\boldsymbol{x}\sim p_{real}} f(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{x}\sim p_{fake}} f(\boldsymbol{x}) \right\|_2^2. \tag{9}$$

Since their development, semi-supervised GANs have been used to improve training in many areas of classification, including digit classification (Springenberg, 2015; Sricharan et al., 2017; Salimans et al., 2016), object classification (Springenberg, 2015; Sricharan et al., 2017; Salimans et al., 2016), facial attribute identification (Sricharan et al., 2017), and image segmentation (per pixel object classification) (Souly et al., 2017).

2.4. Alternative semi-supervised regression GAN methods

For regression, Rezagholiradeh and Haidar (2018) provides two semi-supervised GAN approaches. They have applied their methods to the driving application, which we compare to in Section 4.

First, they present a dual goal GAN (DG-GAN) approach, which they refer to as Reg-GAN Architecture 1. A DG-GAN outputs two labels: a regression value prediction and a fake/real classification prediction. The idea is that the network must learn both how to distinguish between real and fake examples, and how to predict the correct value for a labeled example. However, this approach does not enforce that these two predictions be related. Part of the network may learn the task of identifying real/fake images, while another portion of the network learns the task of predicting regression values. A representation of this split learning can be seen in Fig. 3. If the objective of distinguishing being real and fake examples is weighted strongly enough, the network may devote larger portions of the network to the real/fake classification task, thereby reducing its effectiveness in the regression prediction. We show in our experiments that our proposed method outperforms the DG-GAN, both in our own implementation and in that of Rezagholiradeh and Haidar (2018).

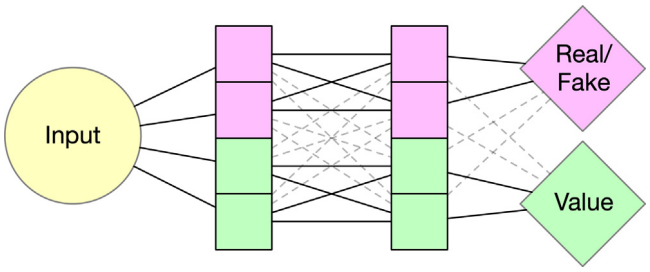


Fig. 3. A DG-GAN network splitting the network into solving the two objectives independently, rather than using a shared representation. The dashed lines represent connections which exist but have very low weights. The degree of this division of learning can vary.

They also present a second which method, Reg-GAN Architecture 2, which only outputs the single driving angle regression value, and then attempts to label this value as fake or real depending on if the value lies within the range of real values from the dataset. This method has two significant limitations. (1) If the full range of the unlabeled dataset is unknown, a correct angle prediction will be incorrectly labeled as fake data. Rezagholiradeh and Haidar (2018) assumes the range of the unlabeled data is known. (2) A bias is introduced, as values near the boundary between fake and real are preferred. This is because a generator which can exactly duplicate unlabeled data will force the discriminator to pick a value on the boundary between fake and real as the best possible answer. Finally, a discrete classification method was presented with each class being the central value of the class interval.

2.5. Regression in conditional GANs

Another distinct category of related work is that of regression in conditional GANs. Conditional GANs are a type of GAN designed to produce realistic examples which have specific desired properties in the example. Bazrafkan and Corcoran (2018) provides an approach to generate images with specific characteristics in a conditional GAN. In particular, they use a regressor in parallel with the discriminator network to provide more variation in the generated examples.

These works are attempting to produce realistic looking generated examples. The produce is not a predictive network for real examples. In contrast, our approach is designed to improve the predictive capabilities of the discriminator on real examples. Notably, *we do not expect our generator to produce realistic looking examples*. On the contrary, we expect the examples generated will not look realistic. As noted by Salimans et al. (2016), the use of feature matching (which is also used in our work) improves discriminator predictive accuracy while reducing the realism of the generated examples. We expect our feature contrasting approach will further erode the realism. Furthermore, works such as Dai et al. (2017) show how a generator which produces examples that are too realistic may be less advantageous for improving a discriminator's predictive abilities.

3. Theory and design

3.1. SR-GAN formulation using feature contrasting

The semi-supervised regression GAN (SR-GAN) approaches regression estimation by comparing the types of available data (labeled,

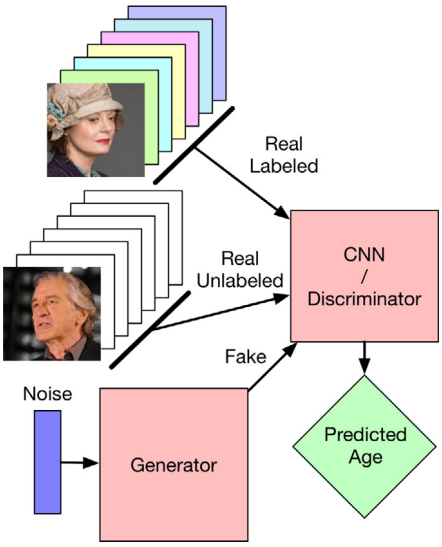


Fig. 4. The structure of an SR-GAN. Its structure is similar to the semi-supervised GAN, with the major differences being in the objective functions and the output being a regression value. In this network, the discriminator distinguishes between fake and real images through feature statistics. No explicit real or fake label is assigned.

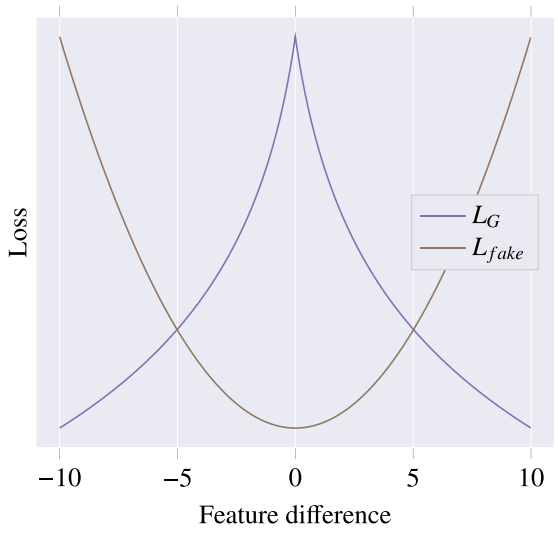


Fig. 5. A comparison of the losses used for feature matching and feature contrasting, used in L_G and $L_{unlabeled}$ respectively. The losses have been normalized for comparison. Shown in the change in loss due to a single feature (due to the norms used in the functions, multiple features changing together have a slightly different impact). Of particular note, a decreased loss for one necessarily results in an increased loss for the other.

unlabeled, and fake) as probability distributions rather than individual examples. In this method, the discriminator does not attempt to predict a label for the unlabeled data or fake data. Instead, the statistics of the features within the network for each type of data is compared. Here is the key idea: We have the discriminator seek to make the unlabeled examples have a similar feature distribution as the labeled examples. The discriminator also works to have fake examples have a feature distribution as divergent from the labeled examples distribution as possible. This forces the discriminator to see both the labeled and unlabeled examples as coming from the same distribution, and fake data as coming from a different distribution. The generator, on the other hand, will be trained to produce examples which match the unlabeled example distribution, and because of this, the generator and discriminator have opposing goals. How a label is assigned to an example drawn from that distribution is still decided by based on the labeled examples (as it is in ordinary DNN/CNN training), but the fact that the unlabeled examples must lie in the true example distribution forces the discriminator to more closely conform to the true underlying data generating distribution. The SR-GAN structure can be seen in Fig. 4 with age estimation as an example. For the case of training the discriminator to have similar feature statistics for both real labeled and real unlabeled data, this approach is related to the feature matching proposed by Salimans et al. (2016), except that this is applied for entirely different purposes than it was in their work. In the case of training the discriminator with real data and fake data, we propose a novel approach, *feature contrasting*, which is antithetical to feature matching. In this case, the discriminator attempts to make the features of the real and fake data as dissimilar as possible, while the generator is attempting to make these features as similar as possible.

Specifically, the loss functions as defined for classification (Eqs. (5) to (7)) in the case of regression will become the following. First, we separate the loss of the discriminator into several terms for clarity. This is given by

$$\begin{aligned} L_D &= L_{supervised} + L_{unsupervised} \\ &= L_{labeled} + L_{unlabeled} + L_{fake} \end{aligned}. \tag{10}$$

What we refer to as the "labeled loss", is given by

$$L_{labeled} = \mathbb{E}_{\boldsymbol{x},y\sim p_{data}(\boldsymbol{x},y)}[(D(\boldsymbol{x}) - y)^2]. \tag{11}$$

This loss is similar to an ordinary fully supervised loss (for regression training). Next, the "unlabeled loss" causes the discriminator to attempt to make the feature statistics of the real labeled data and the real unlabeled data be as similar as possible. This unlabeled loss is given by

$$L_{unlabeled} = \left\| \mathbb{E}_{\boldsymbol{x}\sim p_{labeled}} f(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{x}\sim p_{unlabeled}} f(\boldsymbol{x}) \right\|_2^2. \tag{12}$$

In contrast, the "fake loss" causes to the discriminator to attempt to make the feature statistics of the real data as dissimilar to the fake data as possible. This feature contrasting is accomplished with the loss function given by

$$L_{fake} = -\left\| \log\left(\left| \mathbb{E}_{\boldsymbol{x}\sim p_{fake}} f(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{x}\sim p_{unlabeled}} f(\boldsymbol{x}) \right| + 1 \right) \right\|_1. \tag{13}$$

Finally, the generator attempts to make the feature statistics of the real data match those of the fake data. This goal is accomplished by the generator loss given by

$$L_G = \left\| \mathbb{E}_{\boldsymbol{x}\sim p_{fake}} f(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{x}\sim p_{unlabeled}} f(\boldsymbol{x}) \right\|_2^2. \tag{14}$$

Here, $L_{unlabeled}$ and L_G are identical except in which types of data are being compared. Additionally, the feature contrasting in Eq. (13) is in direct opposition to the feature matching in Eq. (14). Notably, there is no possibility for the generator and discriminator to both benefit by a change in these features; A decreased loss for one necessarily results an increased loss for the other. A comparison of a change in the loss from a single feature can be seen in Fig. 5. We briefly explore some additional loss function options in Section 3.1.

We note that we choose a different norm function for Eq. (13) compared to Eqs. (11) and (14). The L2 norm in Eqs. (11) and (14) causes *any* non-matching feature to be the most heavily punished, resulting in a network which tries to make all features similar. Conversely, an L1 norm is used for feature contrasting. This is because an L2 norm would result in a discriminator which focuses on the already most dissimilar feature while allowing all other features to become similar. The L1 norm puts an equal benefit on contrasting all features. To emphasize this, the L2 norm for L_{fake} results in problematic backpropagation, as zero distance feature differences should result in the largest gradients, but are instead multiplied by zero.

To summarize, the SR-GAN uses feature matching for the discriminator loss functions where in previous methods a separate "fake" class is defined. Specifically this can be seen in the change from

the unsupervised loss in Eq. (7) (which uses a "fake" class in the discriminator) to Eqs. (12) and (13) (which uses feature layer statistics). This accomplishes two goals:

1. Regression problems have no classes and the previous methods require a "fake" class definition, and the SR-GAN approach allows regression problems to be approached.
2. The feature matching does not introduce any bias in the discriminator label prediction, as the final label output is not used in the unsupervised loss.

Additionally, the SR-GAN approach requires no prior information about the data and requires no manual definition of goals beyond the original loss function for labeled examples.

3.2. Gradient penalty

Of the challenges preventing the use of an SR-GAN, the greatest is likely the difficulty of designing an objective which reliably and consistently converges. GANs can easily fail to converge under various circumstances (Barnett, 2018). To solve these general GAN instability issues, we use the gradient penalty approach proposed by Arjovsky et al. (2017) and Gulrajani et al. (2017).

The gradient penalty as defined by Gulrajani et al. (2017) is not applicable to our situation, because their gradient penalty is based on the final output of the discriminator. As the final output of the discriminator is not used in producing the gradient to the generator, we use a modified form of the gradient penalty. This gradient penalty term is added to the rest of the loss function resulting in

$$L = L_{labeled} + L_{unlabeled} + L_{fake} + \lambda \mathbb{E}_{\boldsymbol{x}\sim p_{interpolate}} \left[\max\left(\left(\left\|\nabla_{\hat{\boldsymbol{x}}}(f(\boldsymbol{x}))\right\|_2^2 - 1\right), 0\right)\right]. \tag{15}$$

where $p_{interpolate}$ examples are generated by $\alpha p_{unlabeled} + (1-\alpha)p_{fake}$ for $\alpha \sim \mathcal{U}$. The last term basically provides a restriction on how quickly the discriminator can change relative to the generator's output. Our version of the gradient penalty term is modified in multiple ways from the original. First, as noted above, the final discriminator output cannot be used, nor should it, as the discriminator's interpretation of the generated data only matter in regard to the feature vector, $f(\boldsymbol{x})$. Second, the gradient penalty is normally applied to a term similar to the L_{fake} term using the interpolated values. However, our L_{fake} is based on the average of a batch of fake examples whose difference is then taken from a batch of real examples. As both the L_{fake} term and interpolates are calculated based on the real data, the resulting gradient penalty is negligible. Instead, we apply the gradient penalty directly to the mean feature vector of a batch of interpolated examples and do not apply the feature distance loss function compared to the mean real feature vector. As this penalizes the gradient even for mean feature vectors far from the mean real feature vector, it may slow training. However, near the real feature vector, it approximates the original gradient penalty formulation and works well in practice. Lastly, we use the one-sided version of the gradient penalty described by Gulrajani et al. (2017). As mentioned in their work, the one-sided penalty more closely matches the desired discriminator training properties, and we found this approach to produce higher accuracies than the two-sided penalty.

4. Experiments and results

To demonstrate the capabilities of the semi-supervised regression GANs, we use four experimental setups, each of which consists of several individual trials and demonstrations.

The first experimental setup will be of a synthesized dataset problem. This will allow us to demonstrate the details of the theoretical issues behind a semi-supervised regression GAN in a well controlled and understood environment. These include: what is the right objective which reliably and consistently converges in training, and how little data is needed to achieve different levels of prediction accuracy. We will use a dataset of polynomials with sampled points on the polynomial, whereas the goal of the network is to predict coefficients of the polynomial given the sampled points. Using this simplistic problem, we can show how the semi-supervised regression GAN works in details, what variations can influence its capabilities, and what its limitations are. Most importantly, this allows us to have complete control and understanding of the underlying data generating distribution. This is impossible in any real-world application, as the underlying data generating distribution there is the real world itself.

The downside to the synthetic dataset is that because we have complete control over the data generating distribution, we can define the data such that our SR-GAN does arbitrarily well compared with a normal DNN. As such, the remaining experimental setups are real-world applications. The applications of age estimation, driving steering angle prediction, and crowd counting have been chosen for this purpose. The real world case provides an area we can show direct improvements in compared to a non-adversarial CNN.

4.1. Coefficient estimation

The first experimental setup consists of a simple, well-controlled mathematical model, whose problem can be easily solved with simple neural networks when given enough examples. The example chosen is a polynomial coefficient estimation problem. This problem allows for an environment in which many properties of the semi-supervised regression GAN can be shown and their limits tested. In particular, the simple environment allows us to not only demonstrate the properties of the semi-supervised regression GAN but also give a clear theoretic understanding of why the network exhibits these behaviors. Five important aspects will be discussed: (1) the dataset; (2) the experiment setup; (3) estimation with minimal data; (4) loss function analysis; and (5) choices of gradient penalty.

4.1.1. Polynomial coefficient estimation dataset

For the data of the mathematical model to appropriately represent the characteristics of a real aggression application, we seek to create a data generating model that exhibits the following properties.

1. Able to produce any desired number of examples.
2. The distribution of the underlying data properties is selectable.
3. The relation between the raw data and the label is abstract, where the label is a regression value instead of one of a finite number of classes.
4. Able to contain latent properties that effect the relation between the data and the labels.
5. Most of the data can be made to be irrelevant to the label.

Property 1 allows us to run any number of trials on new data, and run trials where data is unlimited. Property 2 reveals the inner workings of the data distribution. This is important, as we can monitor how closely the generator's examples match the true distribution and examine what kinds of distributions lead to limitations or advantages of the GAN model. Property 3 ensures the findings on the toy model is relevant real deep learning applications for regression. That is, deep learning is typically used in cases where input data is complex, and an abstract, high-level meaning of that data is desired. When the relationship between the data and the label (the regression value) is too simple, more traditional prediction methods tend to be used. Property 4 is also important because of its relationship to real applications. Most applications involve cases where a property which is not the value to be predicted directly effects the data related to value to be predicted. For example, in the case of age estimation, whether the image of the face is lit from the front or lit from the side drastically changes the data and what the CNN should be searching for. Finally, Property 5 requires that our model is able to filter which pieces of information are important

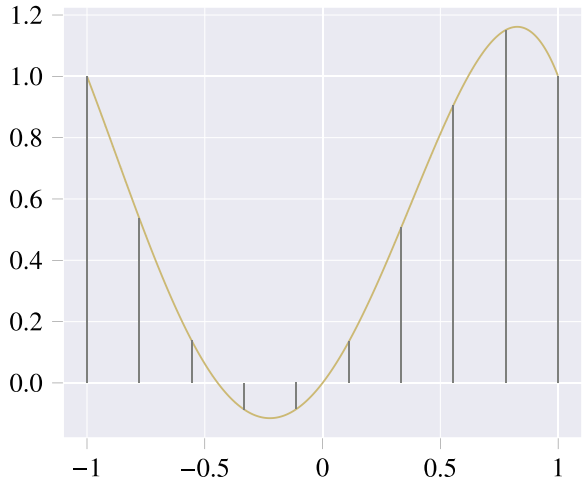


Fig. 6. An example of a polynomial as described in Eq. (16) with 10 points sampled. In this case, $a_2 = 2$, $a_3 = -1$, and $a_4 = -1$, but only a_3 is the coefficient to be estimated.

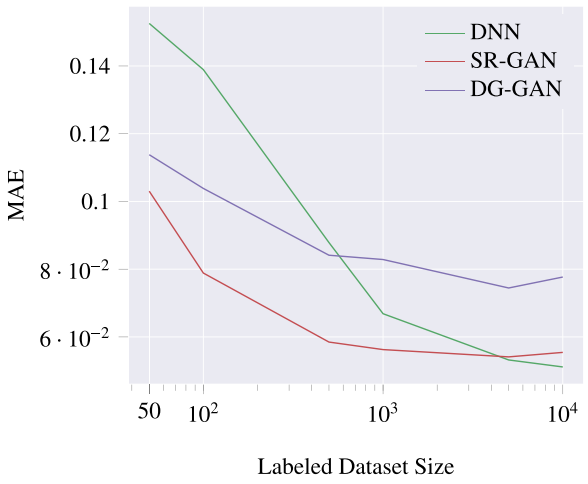


Fig. 7. The resultant inference accuracy of the coefficient estimation network trained with and without the SR-GAN for various quantities of labeled data.

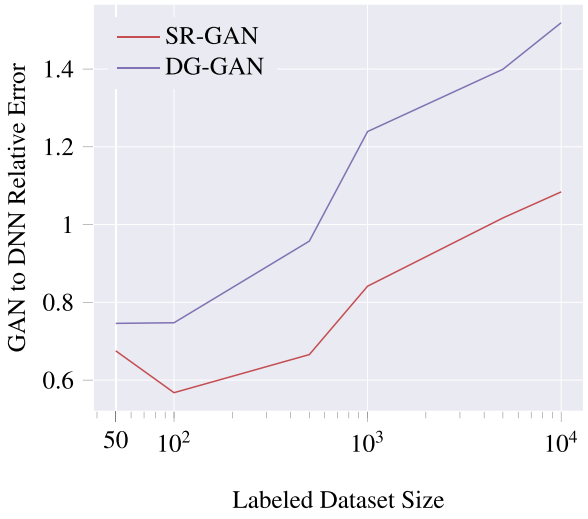


Fig. 8. The relative error of the GAN model over CNN model for various quantities of labeled data for the coefficient model.

and which are not. Again, in the case of age estimation, whether the background behind the person is outdoors or indoors should have little or no impact on the prediction of their age. In many, if not most, cases of deep learning applications the majority of the input data has little to no relevance for the task at hand. The network must learn which information should be relied on and which data should be ignored.

An option of a simplistic mathematical model for this purpose would be a data generating distribution which is defined as follows. First, we define a polynomial,

$$y = a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x. \tag{16}$$

We set $a_1 = 1$. With $\mathcal{U}(r_0, r_1)$ representing a uniform distribution over the range from r_0 to r_1, a_3 is randomly chosen from $\mathcal{U}(-1, 1)$. a_2 and a_4 are randomly chosen from $b \cdot \mathcal{U}(-2, -1) + (1 - b) \cdot \mathcal{U}(1, 2)$ with b being randomly chosen from a standard binomial distribution. Then we sample y for 10 xs from linear space from -1 to 1. An example of such a polynomial and the observed points are shown in Fig. 6. This one polynomial and the observed points constitutes a single example in our dataset. The label of this example we choose as a_3. That is, our network, when given the 10 observations, should be able to predict a_3.

We can compare the pieces of this data generating distribution to the standard image regression problem (think of age estimation from images) to better understand what parts of the toy model represent which parts in a real application model. The 10 observed values from the toy model are analogous to the pixel values in image regression. a_3 is equivalent to the object label (e.g. age value). Finally, the set of all polynomials obtainable from Eq. (16), given the restrictions on how the coefficients are chosen, is the underlying data generating distribution in the toy case, where this role is played by views of the real world projected to an image plane in the regression case (such as age estimation).

This model fulfills all but the last property defined above. To satisfy Property 5, we simply make every example in the dataset consist of 5 different polynomials each chosen and observed as previously explained. However, for this single example (consisting of 5 polynomials) on the a_3 coefficient of the first example is the label. Thus, each example consists of 50 observations, only 10 of which are related to the label. Lastly, we apply noise to every observation.

4.1.2. Coefficient estimation experimental setup

In the coefficient estimation experiments, both the discriminator and generator each consisted of a 4 layer fully connected neural network. Each layer contained 10 hidden units. All code and hyperparameters can be found at https://github.com/golmschenk/srgan. The training dataset for each experiment was randomly chosen. The seed for the random number generator is set to 0 for the first experiment, 1 for the second, and so on. The same seeds are used for each set of experiments. That is, the SR-GAN compared with the DNN use the same training data for each individual trial. Additionally, for experiments over a changing hyperparameter, the same seeds are used for each hyperparameter value.

In these experiments, we demonstrate the value of the SR-GAN on polynomial coefficient estimation. Using a simple fully connected neural network architecture, we have tested the DG-GAN and SR-GAN methods compared to a plain DNN on various quantities of data from the generation process described above. The results of these experiments can be seen in Fig. 7. In each of these experiments an unlabeled dataset of 50,000 examples was used, when various quantities (from 50 to 10,000) of labeled data were used. Each data point on the plots is the average of three training runs randomly seeded to contain different training and test sets on each experiment. The relative error between the DNN and the GAN methods can be seen in Fig. 8. We see a significant accuracy improvement in lower labeled data cases for the GAN methods. The SR-GAN error is 68% of what the DNN error is at with 50 labeled examples. With 50 examples, the DG-GAN also has a significant advantage with 75% the error the DNN has. However, the DG-GAN

Table 1
A comparison of the SR-GAN method using various loss functions for feature matching and feature contrasting. Each experiment was run on the coefficient application with 500 labeled examples and 50,000 unlabeled examples.

Loss functions	MAE
$L_{fake} = -\left\|\log\left(\boldsymbol{d}_f + 1\right)\right\|_1 \quad L_{unlabeled} = L_G = \left\|\boldsymbol{d}_f\right\|_2^2$	0.0578
$L_{fake} = -\left\|\sqrt{\boldsymbol{d}_f + 1}\right\|_1 \quad L_{unlabeled} = L_G = \left\|\boldsymbol{d}_f\right\|_2^2$	0.0613
$L_{fake} = -\left\|\boldsymbol{d}_f\right\|_1 \quad L_{unlabeled} = L_G = \left\|\boldsymbol{d}_f\right\|_2$	0.0672

quickly loses its advantage over the DNN as the data size increases. As the amount of labeled data becomes very large, SR-GAN does not perform better than the DNN. This diminishing return is expected, as we can consider the case of infinite labeled data, where unlabeled data could then provide no additional useful information. We note that for the simple problem of coefficient estimation, 10,000 examples is a very large dataset for training. In each of the real world applications we tested our SR-GAN method in, we did not see a detriment in using the SR-GAN with larger numbers of labeled examples.

4.1.3. Loss function analysis on coefficient estimation

As noted in Section 3.1 we primarily experimented with the loss functions given in Eqs. (11), (13) and (14). However, these are not the only loss functions which could be used for the feature matching and feature contrasting objectives.

We tested three sets of loss functions. We will refer to the feature distance vector as

$$\boldsymbol{d}_f = \left|\mathbb{E}_{\boldsymbol{x}\sim p_1} f(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{x}\sim p_2} f(\boldsymbol{x})\right| \tag{17}$$

where p_1 and p_2 are the appropriate labeled, unlabeled, or fake data distributions depending on if the d_f is being used in the $L_{unlabeled}$, L_{fake}, or L_G terms. With this, we used the feature contrasting and feature matching loss functions given in Table 1. The first is the set of loss functions given previously, which we have already given an explanation for. The second set keeps the same feature matching function but uses a square root as the primary component of the feature contrasting function. This provides a stronger incentive for the discriminator to push features which are already far apart, even further apart. This second approach did slightly worse than the first, likely because focusing on contrasting those features which are most similar between the fake and real examples provides a greater improvement. The third approach uses linear losses. This is similar to the linear fake/real losses used in the WGAN implementation by Arjovsky et al. (2017). The reason for the decreased accuracy is likely the same as for the second case, where features which are already dissimilar are still given too much priority in the feature contrasting.

4.2. Driving steering angle prediction

This application works to predict the steering angle of a car given an image from the front of a car. Such an approach allows for basic partial self-driving/auto-pilot capabilities using a single image (Pan et al., 2017). The dataset (Chen, 2017) consists of 45,567 images from a dashboard-mounted camera, where for each image the current rotation angle of the steering wheel was recorded. The goal of the network is to predict this rotation angle given the front facing view image, whose primary feature is the upcoming road segment.

Rezagholiradeh and Haidar (2018) provides two semi-supervised GAN approaches to train for this application which are described in Section 2.4. Additionally, they also provide a baseline discrete classification method with each class being the central value of the class interval.

Here, we perform the experiments presented by Rezagholiradeh and Haidar (2018) using our SR-GAN approach. In these experiments, varying numbers of labeled images randomly selected from the entire dataset are used for training (up to 7200 images) and testing (9000 images). The remaining images are used as the unlabeled data. We use the DCGAN network architecture (Radford et al., 2015), which matches the architecture presented by Rezagholiradeh and Haidar (2018). This network structure (both generator and discriminator) is shown in Fig. 9. All code and hyperparameters can be found at https://github.com/golmschenk/srgan. We note that we cannot precisely duplicate the experiments by Rezagholiradeh and Haidar (2018), as the images used for training and testing were randomly chosen. We similarly randomly selected our datasets. Our random selections were seeded for reproducibility, and the code at our repository can be used to retrieve the dataset selection for our experiments. Examples of the images, both real and fake, used/generated during training are shown in Fig. 10.

Table 2
Steering angle prediction NAE compared to existing approaches for various amounts of labeled training examples.

Method	100	500	1000	2000	4000	7200
Improved-GAN	–	–	4.38%	4.22%	4.07%	4.06%
Reg-GAN (Arch 1)	–	–	2.43%	2.40%	2.39%	2.36%
Reg-GAN (Arch 2)	–	–	3.81%	3.58%	2.23%	2.21%
SR-GAN	3.12%	2.32%	2.02%	1.89%	1.37%	1.16%

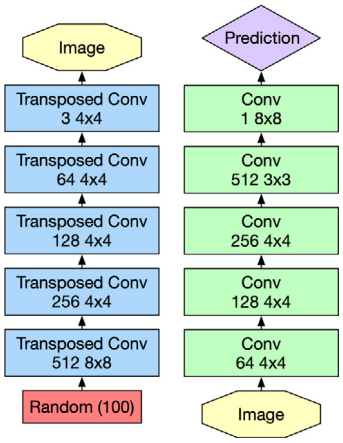


Fig. 9. The DCGAN structure used for the age estimation experiments. The left network is the generator and the right is the discriminator/CNN.

We also note that an entirely random image selection has limited evaluation value for this dataset. The images are part of a video sequence with each image have only minor differences from the previous image. Even a small percentage of the images, when randomly chosen, will contain the primary attributes of a large portion of the dataset. However, for comparison purposes, we have followed the experimental procedure used by Rezagholiradeh and Haidar (2018). We have additionally provided results for significantly lower numbers of labeled images.

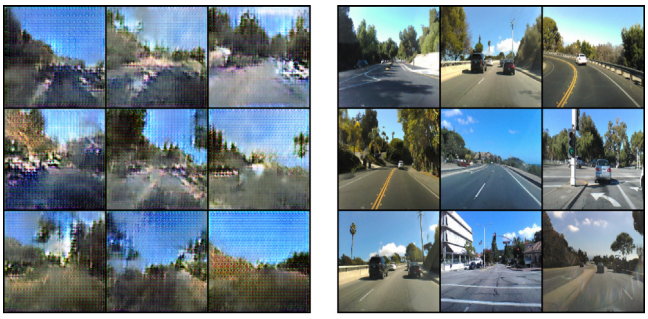
The evaluation metric used is a normalized mean absolute error (NAE) given by

$$NAE = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|y_i - \hat{y}_i\right|}{y_{max} - y_{min}} \times 100\%. \tag{18}$$

The results of our method in comparison to the methods presented by Rezagholiradeh and Haidar (2018) are shown in Table 2. In these experiments, we show that our SR-GAN method significantly outperforms the Reg-GAN method for any number of labeled examples. As Architecture 2 is the more generalized approach of Reg-GAN, it provides the comparison of the most interest.

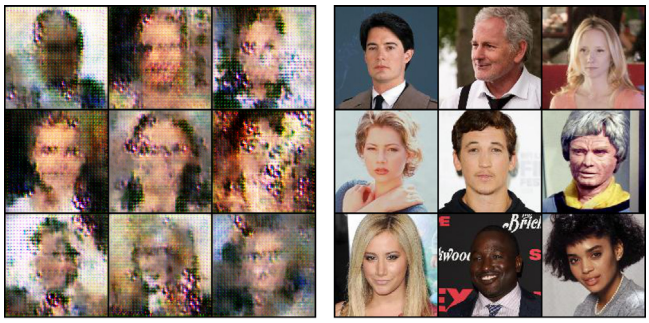
4.3. Age estimation

Age estimation is a well-known regression problem in computer vision using deep learning. In particular, well-established datasets of



(a) Fake steering angle images. (a) Real steering angle images.

Fig. 10. Examples of real and fake images used/generated during training. We note that our approach is not intended to produce realistic looking images, and the fake images are only included for insight.



(a) Fake age images. (a) Real age images.

Fig. 11. Examples of real and fake images used/generated during training. We note that our approach is not intended to produce realistic looking images, and the fake images are only included for insight.

images of individuals with corresponding ages exist and are widely used by the computer vision community. The most notable age estimation database is currently the IMDB-WIKI Face Dataset (Rothe et al., 2016).

For our work, having such a well-known dataset is particularly important as the deep learning community tends to focus on classification problems and not regression problems. Due to this, well-known regression datasets – ones known even outside their domain – tend to be rare. The age estimation dataset is one of these rare cases. It provides a standard which we can test our SR-GAN on which is widely tested on.

4.3.1. Age estimation dataset

The IMDB-WIKI dataset includes over 0.5 million annotated images of faces and the corresponding ages of the people thus imaged. There are 523,051 face images: from 20,284 celebrities, 460,723 face images are from IMDb and 62,328 from Wikipedia. 5% of the celebrities have more than 100 photos, and on average each celebrity has around 23 images.

There are likely many mislabeled images included in the dataset. The image-label pairs were created by searching the top 100,000 actors on IMDb (also known as the "Internet Movie Database").The actors' IMDb profile and Wikipedia page were scraped for images. Face detection was performed on these images, and if a single face detection is found, the image is assumed to be of the correct individual. The image timestamp along with the date of birth of the actor is used to label the image with an age. The image is often a screen capture of a movie, which may have taken years to produce or the screen capture may have happened years later. Additionally, the actor may be purposely made to look a different age in the movie. Despite these many areas of mislabeling, the dataset it thought to consist of overwhelmingly correctly labeled images. To minimize the number of incorrectly labeled images the database is filtered based on several criteria. The database includes face detection scores (certainty of containing a face) and a secondary face score (containing an additional face). If the first face score was too low the image was excluded. If there was a secondary face detected it is also excluded (since these are taken from the actor's IMDb page, it is only assumed to be a picture of the actor if there is only one person in the image). Images labeled with an age below 10 or above 95 are also excluded. Primarily, the below 10 filter is important as many images included an incorrect age of only a few years old. Finally, only images of 256 × 256 resolution or higher are used. After this filtering, we are left with ~90k images. Both the labeled and unlabeled data is taken from these images (without overlap), and the labels were not used for the unlabeled data. Data was selected randomly for each trial. Though other face data could be used for the unlabeled data, for these experiments, we wished to ensure that the labeled and unlabeled data came from the same data distribution (see Figs. 11 and 12).

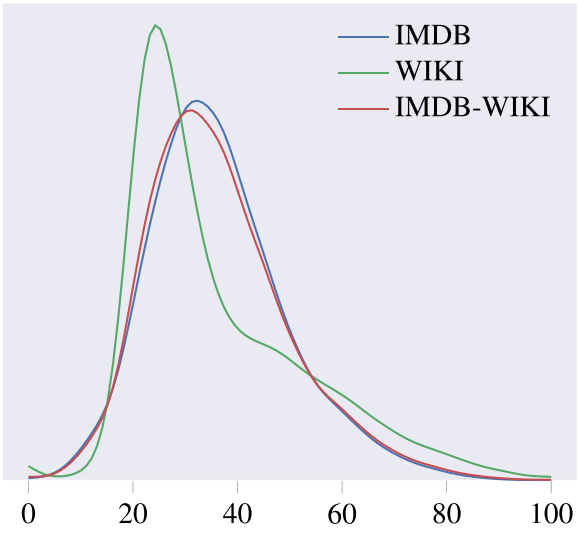


Fig. 12. The distribution of ages in the IMDB-WIKI database.

4.3.2. Age estimation experimental setup

In the age estimation experiments, the DCGAN network architecture (Radford et al., 2015) is used. All code and hyperparameters can be found at https://github.com/golmschenk/srgan. The discriminator of the DCGAN was used alone as the CNN baseline model. The network structure can be seen in Fig. 9. The training dataset for each experiment was randomly chosen. The seed is set to 0 for the first experiment, 1 for the second, and so on. The same seeds are used for each set of experiments. That is, the SR-GAN compared with the CNN use the same training data for each individual trial. This set of experiments used the second set of loss functions from Section 4.1.3.

The following experiments demonstrate the value of the SR-GAN on age estimation. Using a DCGAN (Radford et al., 2015) network architecture, we have tested the SR-GAN method on various quantities of data from the IMDB-WIKI database. The results of these experiments can be seen in Fig. 13. In each of these experiments, an unlabeled dataset of 50,000 images was used, whereas the size of the labeled data samples varies from 10 to 30,000. Each point on this plot is the result of a single randomly seeded training dataset. For each labeled dataset size, 5 trials were run. The relative error between the CNN and the GAN can be seen in Fig. 14. We see a significant accuracy improvement in every case tested. At 100 labeled examples, the GAN achieves a MAE of 10.6, an accuracy which is not achieved by the CNN until it has 5000 labeled examples available for training. At 100 labeled examples, the GAN has 75% the error that the CNN does.

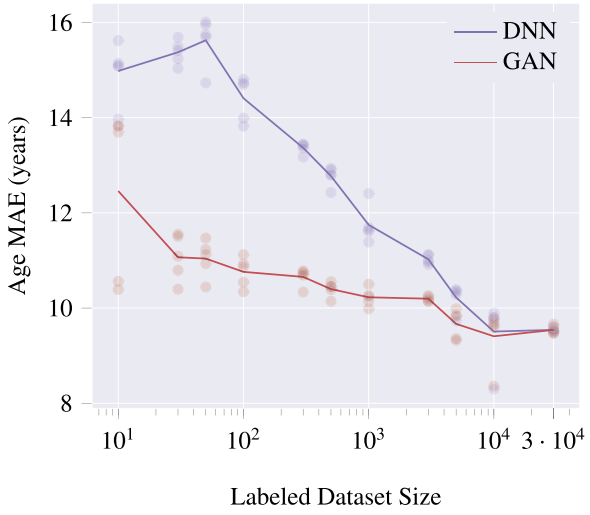


Fig. 13. The resultant inference accuracy of the age estimation network trained with and without the SR-GAN for various quantities of labeled data. Each dot represents a trial with randomized training data, and the line represents the mean of the trials.

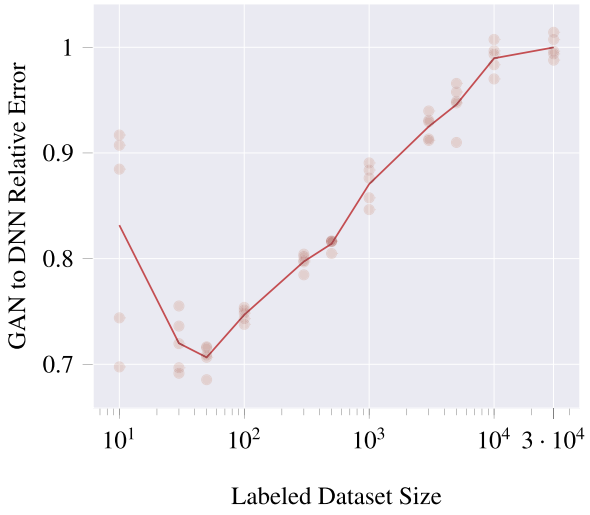


Fig. 14. The relative error of the GAN model over CNN model for various quantities of labeled data for age estimation. Each dot represents a trial with randomized training data, and the line represents the mean of the trials.

The advantage of the SR-GAN drops to near zero as the number of images approaches the number of unlabeled examples being used. There seem to be two likely causes for this. Either, there are enough training images that the network is at capacity (additional images will not further improve the results), or the ratio of labeled to unlabeled images is too small for the generator to be of more benefit to the discriminator. Unfortunately, the number of images available in the IMDB-WIKI dataset make it difficult to pursue a larger number of training examples further.

4.4. Crowd counting

The fourth application we consider is the complex problem of dense crowd counting. Every year, crowds of thousands to millions gather for protests, marathons, pilgrimages, festivals, concerts, and sports events. For each of these events, there is a myriad of reasons to desire to know how many people are present. For those holding the event, both real-time management and future event planning is determined by how many people are present, their current locations, and the intervals at which people are present. For security purposes, evacuations planning and where crowding might be a potential harm to individuals is dependent on the size of the crowds. In journalistic pursuits, the size of a crowd attending an event is often used to measure the significance of the event.

We provide the mean absolute count error (MAE), normalized absolute count error (NAE), and root mean squared count error (RMSE). These are given by the following equations:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{C}_i - C_i\right| \tag{19}$$

$$\text{NAE} = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|\hat{C}_i - C_i\right|}{C_i} \tag{20}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{C}_i - C_i)^2} \tag{21}$$

Idrees et al. (2018) showed that a vanilla DenseNet (Huang et al., 2017) outperformed many application-specific networks for crowd counting. Though Idrees et al. (2018) then provides an application specific version of DenseNet, we chose to use the vanilla version of DenseNet201 as the discriminator in our experiments. This is done to avoid application specific nuances that distract from the main focus of our work, while still providing a network comparable to the state-of-the-art in terms of accuracy. For the generator, we use the same DCGAN generator architecture as was used in our age and steering angle experiments.

The dataset we evaluated our approach on is the ShanghaiTech dataset (Zhang et al., 2016) part A. The dataset is split into two parts, of which we used Part A in our experiments. Part A contains 482 images, 300 for training and 182 for testing. It contains a total of 241,677 head labelings, with an average of 501.4, a maximum of 3139, and a minimum of 33. This part contains a wide range of image sizes, head counts, and perspectives. We used the training and testing images as prescribed by the dataset provider, except we used limited labels for training. A set of example images can be seen in Fig. 15. During the training process, patches of the images are used. During testing, a sliding window approach is used to calculate the count for each patch with overlapping patches being averaged. A final summing of the average values produces a count for the entire image. Examples of the patches, both real and fake, used/generated during training are shown in Fig. 16.

We compare a CNN with the SR-GAN model in our experiments. These results can be seen in Table 3. From the experiments, we can clearly see that the SR-GAN model outperforms the CNN model consistently across various amounts of labeled training images (from 50 to 300), on all three measures. Overall, SR-GAN advantage increases when more training examples are provided. For example, the decreases of MAE of using the SR-GAN versus the CNN are 2.6%, 3.4%, 6.0% to 6.4% for 50, 100, 200 to 300, respectively. This is slightly contrary to what we might expect, as we would assume the advantage of the SR-GAN to diminish as the number of examples becomes very large. However, the increase is small enough that it may simply be due to chance from dataset selection. The percentage in error decreases are small, but they are comparable to the decrease gained by increasing the size of the labeled dataset. In many cases, the SR-GAN provides an improved over the CNN even with smaller numbers of labeled examples. For example, the SR-GAN with 200 images outperforms the CNN with 300 images. Such improvements are also found in the RMSE for the SR-GAN with 100 and 200 labeled examples compared to the CNN with 200 and 300 examples.

Fig. 15. Full image examples from the ShanghaiTech crowd counting dataset.

(a) Fake crowd counting images. (b) Real crowd counting images.

Fig. 16. Examples of real and fake images used/generated during training. We note that our approach is not intended to produce realistic looking images, and the fake images are only included for insight.

Table 3
Crowd counting errors compared various amounts of labeled training examples.

Method	50	100	200	300
CNN MAE	136.9	127.5	119.2	118.0
SR-GAN MAE	133.3	123.2	112.0	110.5
CNN NAE	0.342	0.354	0.359	0.357
SR-GAN NAE	0.339	0.348	0.321	0.323
CNN RMSE	208.5	185.1	183.2	182.3
SR-GAN RMSE	205.9	178.3	178.2	169.5

5. Conclusions

Throughout this work, we have presented a means by which to train semi-supervised GANs in a regression situation. The new SR-GAN algorithm was explained in detail. A set of optimization rules which allows for stable, consistent training when using the SR-GAN, including experiments demonstrating the importance of these rules, were given. We performed systematic experiments using the SR-GAN on the real world applications of age estimation,driving steering angle prediction, and crowd counting, all from single images, showing the benefits of SR-GANs over existing approaches. Adding the SR-GAN generator and objectives to a CNN when unlabeled data is available almost always increases the predictive accuracy of the CNN.

We believe this work demonstrates a way in which semi-supervised GANs can be applied generally to a wide range of regression problems with little or no change to the algorithm presented here. This work allows such problems to be solved using deep learning with significantly less labeled training data than was previously required.

Acknowledgments

This research was initiated under appointments to the U.S. Department of Homeland Security (DHS) Science & Technology Directorate Office of University Programs, administered by the Oak Ridge Institute for Science and Education (ORISE) through an interagency agreement between the U.S. Department of Energy (DOE) and DHS. ORISE is managed by ORAU under DOE contract number 1DE-AC05-06OR23100 and 1DE-SC0014664. All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of DHS, DOE, or ORAU/ORISE. The research is also supported by National Science Foundation through Awards PFI #1827505 and SCC-Planning #1737533, and Bentley Systems, Incorporated, through a CUNY-Bentley Collaborative Research Agreement (CRA). Additional support provided by the Defense Intelligence Agency (DIA) via the Rutgers University Consortium for Critical Technology Studies.

References

Ali, I., Greifeneder, F., Stamenkovic, J., Neumann, M., Notarnicola, C., 2015. Review of machine learning approaches for biomass and soil moisture retrievals from remote sensing data. Remote Sens. 7 (12), 16398–16421.

Arjovsky, M., Chintala, S., Bottou, L., 2017. Wasserstein gan, arXiv preprint arXiv:1701.07875.

Barnett, S.A., 2018. Convergence Problems with Generative Adversarial Networks (GANs), arXiv preprint arXiv:1806.11382.

Bazrafkan, S., Corcoran, P., 2018. Versatile Auxiliary Regressor with Generative Adversarial network (VAR+ GAN), arXiv preprint arXiv:1805.10864.

Bland, L.M., Collen, B., Orme, C.L., Bielby, J., 2015. Predicting the conservation status of data-deficient species. Conserv. Biol. 29 (1), 250–259.

Chen, S., 2017. Sully Chen Driving Dataset. https://github.com/SullyChen/driving-datasets [Online; accessed 8-February-2019].

Dai, Z., Yang, Z., Yang, F., Cohen, W.W., Salakhutdinov, R.R., 2017. Good semi-supervised learning that requires a bad gan. In: Advances in Neural Information Processing Systems. pp. 6513–6523.

Ding, X., Zhang, Y., Liu, T., Duan, J., 2015. Deep learning for event-driven stock prediction.. In: Ijcai. pp. 2327–2333.

Dodge, S., Karam, L., 2017. A Study and Comparison of Human and Deep Learning Recognition Performance Under Visual Distortions, arXiv preprint arXiv:1705.02498.

Eigen, D., Puhrsch, C., Fergus, R., 2014. Depth map prediction from a single image using a multi-scale deep network. In: Advances in Neural Information Processing Systems. pp. 2366–2374.

Elsayed, G., Shankar, S., Cheung, B., Papernot, N., Kurakin, A., Goodfellow, I., Sohl-Dickstein, J., 2018. Adversarial examples that fool both computer vision and time-limited humans. In: Advances in Neural Information Processing Systems. pp. 3914–3924.

Fabbro, S., Venn, K., O'Briain, T., Bialek, S., Kielty, C., Jahandar, F., Monty, S., 2017. An application of deep learning in the analysis of stellar spectra. Mon. Not. R. Astron. Soc..

Fefferman, C., Mitter, S., Narayanan, H., 2016. Testing the manifold hypothesis. J. Amer. Math. Soc. 29 (4), 983–1049.

Goodfellow, I., 2016. NIPS 2016 tutorial: Generative adversarial networks, arXiv preprint arXiv:1701.00160.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets. In: Advances in Neural Information Processing Systems. pp. 2672–2680.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C., 2017. Improved training of wasserstein gans. In: Advances in Neural Information Processing Systems. pp. 5769–5779.

Hartikainen, J., Seppanen, M., Sarkka, S., 2012. State-space inference for non-linear latent force models with application to satellite orbit prediction, arXiv preprint arXiv:1206.4670.

Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q., 2017. Densely connected convolutional networks.. In: CVPR, Vol. 1. p. 3.

Idrees, H., Tayyab, M., Athrey, K., Zhang, D., Al-Maadeed, S., Rajpoot, N., Shah, M., 2018. Composition Loss for Counting, Density Map Estimation and Localization in Dense Crowds, arXiv preprint arXiv:1808.01050.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.

LeCun, Y., Haffner, P., Bottou, L., Bengio, Y., 1999. Object recognition with gradient-based learning, Shape, contour and grouping in computer vision, 823–823.

Liu, Y., Weisberg, R.H., 2005. Patterns of ocean current variability on the west florida shelf using the self-organizing map. J. Geophys. Res.: Oceans 110 (C6).
Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F.-Y., 2015. Traffic flow prediction with big data: a deep learning approach. IEEE Trans. Intell. Transp. Syst. 16 (2), 865–873.
Marino, D.L., Amarasinghe, K., Manic, M., 2016. Building energy load forecasting using deep neural networks. In: Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE. IEEE, pp. 7046–7051.
Niu, Z., Zhou, M., Wang, L., Gao, X., Hua, G., 2016. Ordinal regression with multiple output cnn for age estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4920–4928.
Oliveira, T.P., Barbar, J.S., Soares, A.S., 2016. Computer network traffic prediction: a comparison between traditional and deep learning neural networks. Int. J. Big Data Intell. 3 (1), 28–37.
Pan, X., You, Y., Wang, Z., Lu, C., 2017. Virtual to real reinforcement learning for autonomous driving, arXiv preprint arXiv:1704.03952.
Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A., 2016. Context encoders: Feature learning by inpainting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2536–2544.
Radford, A., Metz, L., Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv preprint arXiv:1511.06434.
Rezagholiradeh, M., Haidar, M.A., 2018. Reg-gan: Semi-supervised learning based on generative adversarial networks for regression. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 2806–2810.
Rothe, R., Timofte, R., Van Gool, L., 2016. Deep expectation of real and apparent age from a single image without facial landmarks. Int. J. Comput. Vis. 1–14.
Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., 2016. Improved techniques for training gans. In: Advances in Neural Information Processing Systems. pp. 2234–2242.
Schwarz, M., Schulz, H., Behnke, S., 2015. RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, pp. 1329–1335.
Souly, N., Spampinato, C., Shah, M., 2017. Semi and Weakly Supervised Semantic Segmentation Using Generative Adversarial Network, arXiv preprint arXiv:1703.09695.
Springenberg, J.T., 2015. Unsupervised and semi-supervised learning with categorical generative adversarial networks, arXiv preprint arXiv:1511.06390.
Sricharan, K., Bala, R., Shreve, M., Ding, H., Saketh, K., Sun, J., 2017. Semi-supervised conditional gans, arXiv preprint arXiv:1708.05789.
Xingjian, S., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., Woo, W.-c., 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: Advances in Neural Information Processing Systems. pp. 802–810.
Zhang, C., Li, H., Wang, X., Yang, X., 2015. Cross-scene crowd counting via deep convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 833–841.
Zhang, Y., Zhou, D., Chen, S., Gao, S., Ma, Y., 2016. Single-image crowd counting via multi-column convolutional neural network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 589–597.

Greg Olmschenk is a PhD candidate in Computer Science at the Graduate Center of the City University of New York. His focus is deep neural networks, particularly generative adversarial networks and methods used for computer vision applications.

Zhigang Zhu received his BE, ME and PhD degrees, all in computer science, from Tsinghua University, Beijing. He is Herbert G. Kayser Chair Professor of Computer Science, at The City College of New York (CCNY) and The CUNY Graduate Center, where he directs the City College Visual Computing Laboratory (CcvcL). His research interests include 3D computer vision, multimodal sensing, virtual/augmented reality, and various applications in assistive technology, robotics, surveillance and transportation. He has published over 150 technical papers in the related fields. He is an Associate Editor of the Machine Vision Applications Journal, Springer, and the IFAC Mechatronics Journal, Elsevier.

Hao Tang is an Associate Professor of Computer Science at The Borough of Manhattan Community College, CUNY. He earned his Ph.D. degree in Computer Science, concentrating in the Computer Vision, at the Graduate Center of CUNY. His research interests are in the fields of 3D computer modeling, HCI, mobile vision and navigation and the applications in surveillance, assistive technology, and education. His research paper was selected as the best paper finalist of International Conference on Multimedia and Expo.