Conditional Generative Adversarial Nets

Akilesh B, CS13B1042

Indian Institute of Technology, Hyderabad

May 3, 2016

What is GAN?

- Generative adversarial networks are a recently introduced method for training generative models with neural networks.
- This approach sidesteps some of the common problems among generative models and adopts a simple SGD training regime.
- The generative model yielded by a learned GAN can easily serve as a density model of the training data.
- Sampling is simple and efficient: the network accepts some noise as input and outputs new samples of data in line with the observed training data.

What is cGAN?

- A conditional generative adversarial network (hereafter cGAN) is a simple extension of the basic GAN model which allows the model to condition on external information.
- This makes it possible to engage the learned generative model in different modes by providing it with different contextual information.

Objective

- Use the cGAN network to generate faces on LFWCrop dataset.
- Compare results of GAN and cGAN.

Dataset

- Cropped version of The Labeled Faces in the Wild images dataset [3] consisting of 13000 color images, known as LFWcrop. The cropping is done to avoid noisy background data.
- The cropped faces in LFWcrop exhibit real-life conditions, including misalignment, scale variations, in-plane as well as out-of-plane rotations.
- Each image has confidence values for a large number of facial expression attributes and related features, which is exploited as conditional data y in the experiments.

Study methodology

- The GAN framework establishes two distinct players, a generator G and discriminator D, and poses the two in an adversarial game.
- The discriminator is tasked with distinguishing between samples from the model and samples from the training data; at the same time, the generator is tasked with maximally confusing the discriminator.
- G and D are both trained simultaneously: we adjust parameters for G to minimize $log(1 - D(G(z))$ and adjust parameters for D to maximize $log(D(X))$.

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{z}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Study methodology

- In cGAN both the generator and discriminator are conditioned on some extra information y.
- y could be any kind of auxiliary information, such as class labels or data from other modalities.
- Conditioning can be performed by feeding y into the both the discriminator and generator as additional input layer.
- In the generator the prior input noise $p_z(z)$, and y are combined in joint hidden representation, and the adversarial training framework allows for considerable flexibility in how this hidden representation is composed.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]$$

Training

The training process for the above model:

- The generator outputs random RGB noise by default.
- The discriminator learns basic convolutional filters in order to distinguish between face images and random noise.
- The generator learns the correct bias (skin tone) and basic filters to confuse the discriminator.
- The discriminator becomes more attuned to real facial features in order to distinguish between the simple trick images from the generator and real face images. Furthermore, the discriminator learns to use signals in the conditional data y to look for particular triggers in the image.

Training

- This process continues until the discriminator is maximally confused.
- Since the discriminator outputs the probability that an input image was sampled from the training data, we would expect a maximally confused discriminator to consistently output a probability of 0.5 for inputs both from the training data and from the generator.

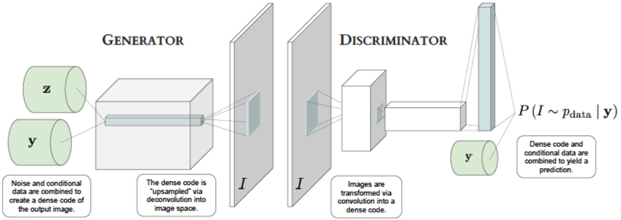
Conditional sampling

- We need to sample images from the generator at training time in order to evaluate the two players.
- The sampling requires both noise z and conditional data y. The random noise can be easily sampled, but care needs to be taken about generating conditional data.
- In case, we draw these directly from the training examples, the generator might be able to reach some spurious optimum where it learns to reproduce each training image based on the conditional data input.

Conditional sampling

- In order to avoid this, conditional data sampling is randomized during training.
- A kernel density estimate $p_y(y)$ (also known as a Parzen window estimate) is built using the conditional values $\{y_i\}_{i=1}^{n}$ drawn from the training data.
- A Gaussian kernel is used, and cross-validate the kernel width σ using a held-out validation set. Samples from this nonparametric density model are used as the inputs to the generator during training.

Model Architecture



Noise and conditional data are combined to create a dense code of the output image.

The dense code is "upsampled" via deconvolution into image space.

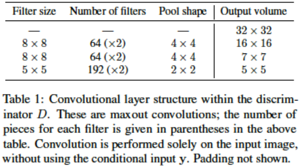
Images are transformed via convolution into a dense code.

Dense code and conditional data are combined to yield a prediction.

Model Architecture

- The generator G is a deconvolutional neural network which runs filters over its inputs and expands rather than contracts the inputs into a new representation of higher spatial dimension.
- This deconvolution architecture was successfully used by Goodfellow et al. [2] to reconstruct CIFAR-10 images.
- The deconvolutional forward pass is calculated just as is the backward pass of a convolutional layer.
- They run just a single deconvolution in this model.

Model Architecture



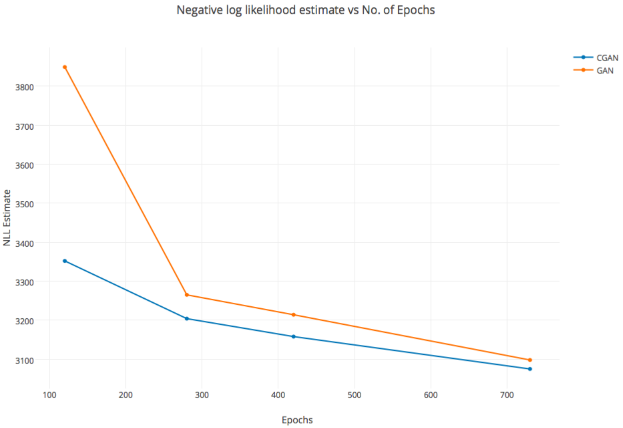
Filter size	Number of filters	Pool shape	Output volume
—	—	—	32×32
8×8	64 (×2)	4×4	16×16
8×8	64 (×2)	4×4	7×7
5×5	192 (×2)	2×2	5×5

Table 1: Convolutional layer structure within the discriminator D. These are maxout convolutions; the number of pieces for each filter is given in parentheses in the above table. Convolution is performed solely on the input image, without using the conditional input $\mathbf{y}$. Padding not shown.

- The discriminator D is a familiar convolutional neural network, similar to any recent model used in discriminative vision tasks such as image classification.
- Each convolutional layer has maxout activations.
- We treat the final output of the convolutions as a dense code describing the input image.

Results

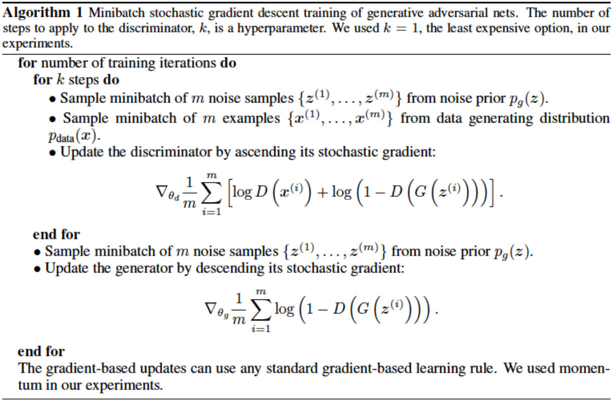
Log Likelihood Comparision



Conclusion

- An extension of the GAN framework with ability to condition on arbitrary external information to both the generator and discriminator components was studied.
- The development of a deterministic control slot in the GAN model opens up exciting possibilities for new models and applications.
- For example, a cGAN could easily accept a multimodal embedding as conditional input y.
- This y could be produced by a neural language model, allowing us to generate images from spoken or written descriptions of their content.

Training of Generator



Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```
for number of training iterations do
  for $k$ steps do
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Sample minibatch of $m$ examples $\{x^{(1)}, \dots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
    • Update the discriminator by ascending its stochastic gradient:
          $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \right].$$
  end for
  • Sample minibatch of $m$ noise samples $\{z^{(1)}, \dots, z^{(m)}\}$ from noise prior $p_g(z)$.
  • Update the generator by descending its stochastic gradient:
          $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$$
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.
```

Training of Generator

- We can see from the algorithm that Discriminator is updated before Generator in each iteration.
- So, according to the algorithm, to update generator, we need the output value when image generated by generator is passed through the discriminator.
- So, we don't need any labels to train generator.
- This is our understanding of the training of generator.

Regarding Minimax objective function

- In practice, the minimax equation described earlier may not provide sufficient gradient for G to learn well.
- Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data.
- In this case, $log(1 - D(G(z)))$ saturates.
- Rather than training G to minimize $log(1 - D(G(z)))$ we can train G to maximize $log(D(G(z)))$.
- This objective function results in the same fixed point of the dynamics of G and D but provides much stronger gradients early in learning.

Our thoughts about the paper

- An extension of GAN framework with the ability to condition on arbitrary external information was added to both the generator and discriminator components.
- The development of a deterministic control slot in the GAN model opens up exciting possibilities for new models and applications.
- For example, a cGAN could easily accept a multimodal embedding as conditional input y.
- This y could be produced by a neural language model, allowing us to generate images from spoken or written descriptions of their content.
- The neural network starts dreaming and we can sample brand-new faces from the learned density model.
- The model learns what a face looks like and learns to draw new ones from scratch which don't resemble faces in the training data provided to the model

References

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and Composing Robust Features with Denoising Autoencoders. In Proceedings of the 25th International Conference on Machine Learning, ICML 08, pages 10961103, New York, NY, USA, 2008. ACM.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. WardeFarley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In Advances in Neural Information Process- ing Systems 27, pages 26722680. Curran Associates, Inc., 2014.

G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recog- nition in unconstrained environments. Technical report, Tech- nical Report 07-49, University of Massachusetts, Amherst, 2007.