A.I. Wiki

A Beginner's Guide to Generative Adversarial Networks (GANs)

You might not think that programmers are artists, but programming is an extremely creative profession. It's logic-based creativity. - John Romero

Generative adversarial networks (GANs) are deep neural net architectures comprised of two nets, pitting one against the other (thus the "adversarial").

GANs were introduced in a paper by Ian Goodfellow and other researchers at the University of Montreal, including Yoshua Bengio, in 2014. Referring to GANs, Facebook's AI research director Yann LeCun called adversarial training "the most interesting idea in the last 10 years in ML."

GANs' potential is huge, because they can learn to mimic any distribution of data. That is, GANs can be taught to create worlds eerily similar to our own in any domain: images, music, speech, prose. They are robot artists in a sense, and their output is impressive – poignant even.

In a surreal turn, Christie's sold a portrait for $432,000 that had been generated by a GAN, based on open-source code written by Robbie Barrat of Stanford. Like most true artists, he didn't see any of the money, which instead went to the French company, Obvious.[0]

Generative vs. Discriminative Algorithms

To understand GANs, you should know how generative algorithms work, and for that, contrasting them with discriminative algorithms is instructive. Discriminative algorithms try to classify input data; that is, given the features of an instance of data, they predict a label or category to which that data belongs.

For example, given all the words in an email (the data instance), a discriminative algorithm could predict whether the message is `spam` or `not_spam`. `spam` is one of the labels, and the bag of words gathered from the email are the features that constitute the input data. When this problem is expressed mathematically, the label is called `y` and the features are called `x`. The formulation `p(y|x)` is used to mean “the probability of y given x”, which in this case would translate to “the probability that an email is spam given the words it contains.”

So discriminative algorithms map features to labels. They are concerned solely with that correlation. One way to think about generative algorithms is that they do the opposite. Instead of predicting a label given certain features, they attempt to predict features given a certain label.

The question a generative algorithm tries to answer is: Assuming this email is spam, how likely are these features? While discriminative models care about the relation between `y` and `x`, generative models care about “how you get x.” They allow you to capture `p(x|y)`, the probability of `x` given `y`, or the probability of features given a label or category. (That said, generative algorithms can also be used as classifiers. It just so happens that they can do more than categorize input data.)

Another way to think about it is to distinguish discriminative from generative like this:

- Discriminative models learn the boundary between classes
- Generative models model the distribution of individual classes

How GANs Work

One neural network, called the *generator*, generates new data instances, while the other, the *discriminator*, evaluates them for authenticity; i.e. the discriminator decides whether each instance of data that it reviews belongs to the actual training dataset or not.

Let's say we're trying to do something more banal than mimic the Mona Lisa. We're going to generate hand-written numerals like those found in the MNIST dataset, which is taken from the real world. The goal of the discriminator, when shown an instance from the true MNIST dataset, is to recognize those that are authentic.

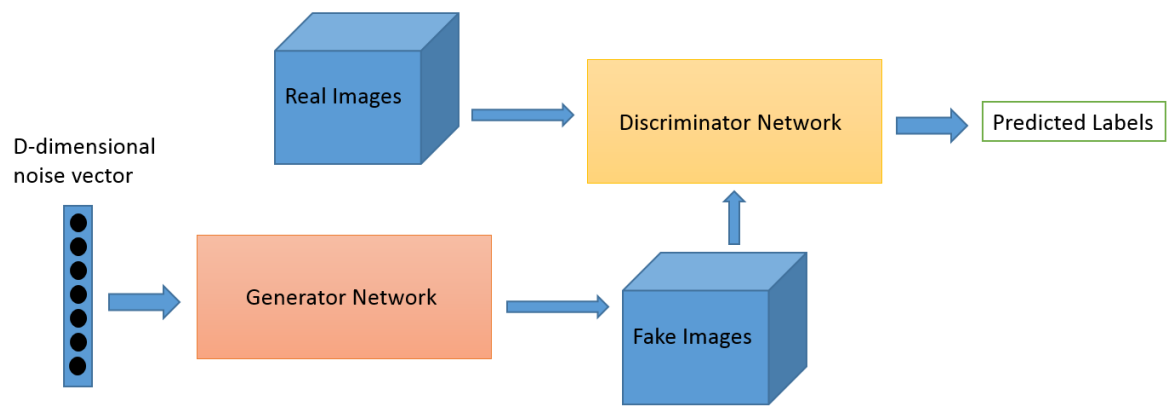
Meanwhile, the generator is creating new, synthetic images that it passes to the discriminator. It does so in the hopes that they, too, will be deemed authentic, even though they are fake. The goal of the generator is to generate passable hand-written digits: to lie without being caught. The goal of the discriminator is to identify images coming from the generator as fake.

Here are the steps a GAN takes:

- The generator takes in random numbers and returns an image.
- This generated image is fed into the discriminator alongside a stream of images taken from the actual, ground-truth dataset.
- The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

So you have a double feedback loop:

- The discriminator is in a feedback loop with the ground truth of the images, which we know.
- The generator is in a feedback loop with the discriminator.



Credit: O'Reilly

You can think of a GAN as the opposition of a counterfeiter and a cop in a game of cat and mouse, where the counterfeiter is learning to pass false notes, and the cop is learning to detect them. Both are dynamic; i.e. the cop is in training, too (to extend the analogy, maybe the central bank is flagging bills that slipped through), and each side comes to learn the other's methods in a constant escalation.

For MNIST, the discriminator network is a standard convolutional network that can categorize the images fed to it, a binomial classifier labeling images as real or fake. The generator is an inverse convolutional network, in a sense: While a standard convolutional classifier takes an image and downsamples it to produce a probability, the generator takes a vector of random noise and upsamples it to an image. The first throws away data through downsampling techniques like maxpooling, and the second generates new data.

Both nets are trying to optimize a different and opposing objective function, or loss function, in a zero-zum game. This is essentially an actor-critic model. As the discriminator changes its behavior, so does the generator, and vice versa. Their losses push against each other.

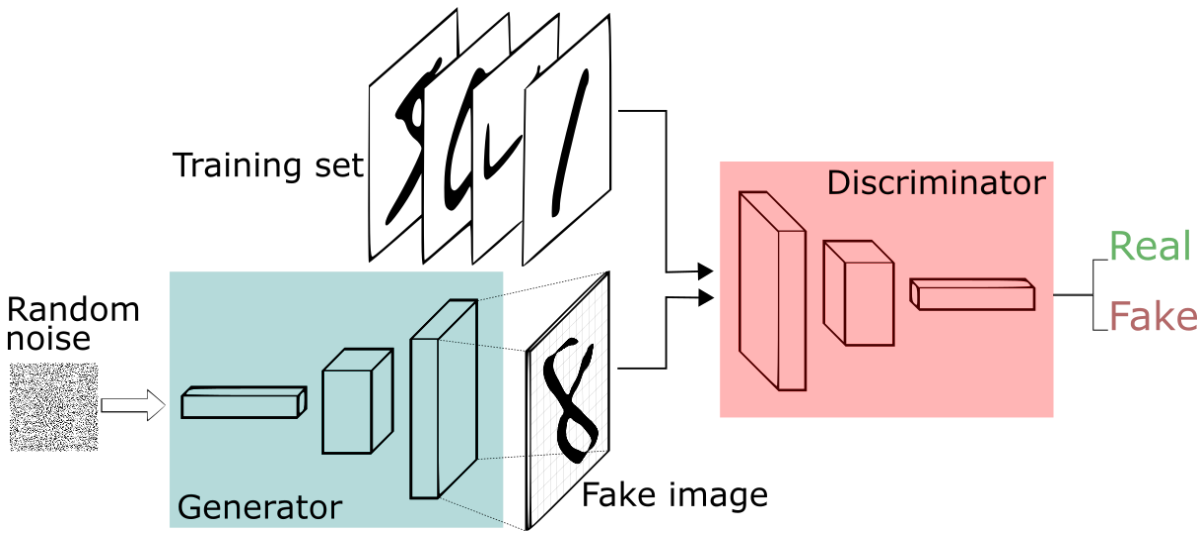


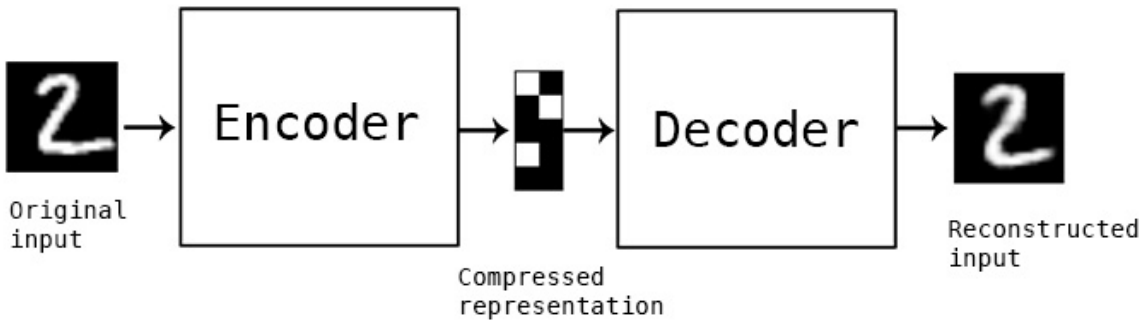
Image credit: Thalles Silva

If you want to learn more about generating images, Brandon Amos wrote a great post about interpreting images as samples from a probability distribution.

GANs, Autoencoders and VAEs

It may be useful to compare generative adversarial networks to other neural networks, such as autoencoders and variational autoencoders.

Autoencoders *encode* input data as vectors. They create a hidden, or compressed, representation of the raw data. They are useful in dimensionality reduction; that is, the vector serving as a hidden representation compresses the raw data into a smaller number of salient dimensions. Autoencoders can be paired with a so-called decoder, which allows you to reconstruct input data based on its hidden representation, much as you would with a restricted Boltzmann machine.



Credit: Keras blog

Variational autoencoders are generative algorithm that add an additional constraint to encoding the input data, namely that the hidden representations are normalized. Variational autoencoders are capable of both compressing data like an autoencoder and

synthesizing data like a GAN. However, while GANs generate data in fine, granular detail, images generated by VAEs tend to be more blurred. Deeplearning4j's examples include both autoencoders and variational autoencoders.

You can bucket generative algorithms into one of three types:

- Given a label, they predict the associated features (Naive Bayes)
- Given a hidden representation, they predict the associated features (VAE, GAN)
- Given some of the features, they predict the rest (inpainting, imputation)

Tips in Training a GAN

When you train the discriminator, hold the generator values constant; and when you train the generator, hold the discriminator constant. Each should train against a static adversary. For example, this gives the generator a better read on the gradient it must learn by.

By the same token, pretraining the discriminator against MNIST before you start training the generator will establish a clearer gradient.

Each side of the GAN can overpower the other. If the discriminator is too good, it will return values so close to 0 or 1 that the generator will struggle to read the gradient. If the generator is too good, it will persistently exploit weaknesses in the discriminator that lead to false negatives. This may be mitigated by the nets' respective learning rates. The two neural networks must have a similar "skill level." [1]

GANs take a long time to train. On a single GPU a GAN might take hours, and on a single CPU more than a day. While difficult to tune and therefore to use, GANs have stimulated a lot of interesting research and writing.

Just Show Me the Code

Here's an example of a GAN coded in Keras, from which models can be imported to Deeplearning4j.

```
class GAN():
    def __init__(self):
        self.img_rows = 28
        self.img_cols = 28
        self.channels = 1
        self.img_shape = (self.img_rows, self.img_cols, self.channels)

        optimizer = Adam(0.0002, 0.5)

        # Build and compile the discriminator
        self.discriminator = self.build_discriminator()
        self.discriminator.compile(loss='binary_crossentropy',
            optimizer=optimizer,
            metrics=['accuracy'])

        # Build and compile the generator
        self.generator = self.build_generator()
        self.generator.compile(loss='binary_crossentropy',
optimizer=optimizer)

        # The generator takes noise as input and generated imgs
        z = Input(shape=(100,))
        img = self.generator(z)

        # For the combined model we will only train the generator
        self.discriminator.trainable = False

        # The valid takes generated images as input and determines
validity
        valid = self.discriminator(img)

        # The combined model  (stacked generator and discriminator)
takes
        # noise as input => generates images => determines validity
        self.combined = Model(z, valid)
        self.combined.compile(loss='binary_crossentropy',
optimizer=optimizer)

    def build_generator(self):

        noise_shape = (100,)

        model = Sequential()

        model.add(Dense(256, input_shape=noise_shape))
        model.add(LeakyReLU(alpha=0.2))
        model.add(BatchNormalization(momentum=0.8))
        model.add(Dense(512))
        model.add(LeakyReLU(alpha=0.2))
        model.add(BatchNormalization(momentum=0.8))
```

```
        model.add(Dense(1024))
        model.add(LeakyReLU(alpha=0.2))
        model.add(BatchNormalization(momentum=0.8))
        model.add(Dense(np.prod(self.img_shape), activation='tanh'))
        model.add(Reshape(self.img_shape))

        model.summary()

        noise = Input(shape=noise_shape)
        img = model(noise)

        return Model(noise, img)

    def build_discriminator(self):

        img_shape = (self.img_rows, self.img_cols, self.channels)

        model = Sequential()

        model.add(Flatten(input_shape=img_shape))
        model.add(Dense(512))
        model.add(LeakyReLU(alpha=0.2))
        model.add(Dense(256))
        model.add(LeakyReLU(alpha=0.2))
        model.add(Dense(1, activation='sigmoid'))
        model.summary()

        img = Input(shape=img_shape)
        validity = model(img)

        return Model(img, validity)

    def train(self, epochs, batch_size=128, save_interval=50):

        # Load the dataset
        (X_train, _), (_, _) = mnist.load_data()

        # Rescale -1 to 1
        X_train = (X_train.astype(np.float32) - 127.5) / 127.5
        X_train = np.expand_dims(X_train, axis=3)

        half_batch = int(batch_size / 2)

        for epoch in range(epochs):

            # ---------------------
            #  Train Discriminator
            # ---------------------

            # Select a random half batch of images
```

```
            idx = np.random.randint(0, X_train.shape[0], half_batch)
            imgs = X_train[idx]

            noise = np.random.normal(0, 1, (half_batch, 100))

            # Generate a half batch of new images
            gen_imgs = self.generator.predict(noise)

            # Train the discriminator
            d_loss_real = self.discriminator.train_on_batch(imgs,
np.ones((half_batch, 1)))
            d_loss_fake = self.discriminator.train_on_batch(gen_imgs,
np.zeros((half_batch, 1)))
            d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)



            # ---------------------
            #  Train Generator
            # ---------------------

            noise = np.random.normal(0, 1, (batch_size, 100))

            # The generator wants the discriminator to label the
generated samples
            # as valid (ones)
            valid_y = np.array([1] * batch_size)

            # Train the generator
            g_loss = self.combined.train_on_batch(noise, valid_y)

            # Plot the progress
            print ("%d [D loss: %f, acc.: %.2f%%] [G loss: %f]" %
(epoch, d_loss[0], 100*d_loss[1], g_loss))

            # If at save interval => save generated image samples
            if epoch % save_interval == 0:
                self.save_imgs(epoch)

    def save_imgs(self, epoch):
        r, c = 5, 5
        noise = np.random.normal(0, 1, (r * c, 100))
        gen_imgs = self.generator.predict(noise)

        # Rescale images 0 - 1
        gen_imgs = 0.5 * gen_imgs + 0.5

        fig, axs = plt.subplots(r, c)
        cnt = 0
        for i in range(r):
            for j in range(c):
```

```
                axs[i,j].imshow(gen_imgs[cnt, :,:,0], cmap='gray')
                axs[i,j].axis('off')
                cnt += 1
        fig.savefig("gan/images/mnist_%d.png" % epoch)
        plt.close()



if __name__ == '__main__':
    gan = GAN()
    gan.train(epochs=30000, batch_size=32, save_interval=200)
```

"They don't appear to want to take over. They just want to dance."

Credit: The New Yorker

Footnotes

0) *Students of the history of the French technology sector should ponder why this is one of the few instances when the French have shown themselves more gifted at marketing technology than at making it. France, a country with strong math pedagogy yet surprisingly Luddite tendencies in wider society, tends to build tech better than they market it. Why didn't Minitel take over the world? Why did Jean-Louis Gassée and countless others feel it was necessary to quit France for America or London?*

1) *It's interesting to consider evolution in this light, with genetic mutation on the one hand, and natural selection on the other, acting as two opposing algorithms within a larger process. What we are witnessing during the Anthropocene is the victory of one half of the evolutionary algorithm over the other; i.e. the genetic mutations in one species, homo sapiens, have enabled the creation of tools so powerful that natural selection plays very little part in shaping us. Some might speculate that that imbalance is leading to a catastrophic collapse of the system, much as we see with poorly tuned GANs. To take it a step further, perhaps this is the structural flaw in the development of intelligent life, akin to a Great Filter, which explains why humans have not found signs of other advanced species in the universe, despite the mathematical probability that such life should arise in a universe so large.*

Homo sapiens is evolving faster than other species we compete with for resources. Along those lines, we might entertain a definition of intelligence that is primarily about speed. All other things being equal, the more intelligent organism (or species or algorithm) solves the same problem in less time. What we see now in the field of AI is an acceleration of algorithms' ability to solve an increasing number of problems, boosted by faster chips, parallel computation, and hundreds of millions in research funding. Elon Musk has expressed his concern about AI, but he has not expressed that concern simply enough, based on a clear analogy. Algorithms are learning faster than we are, just as we learn faster than the species we are driving to extinction. It's about speed. Massively parallelized hardware is a way of parallelizing time. And that is something that the human brain can not yet benefit from.

Resources for Generative Networks

- "Generative Learning algorithms" - Andrew Ng's Stanford notes
- On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes, by Andrew Ng and Michael I. Jordan
- The Math Behind Generative Adversarial Networks
- A Style-Based Generator Architecture for Generative Adversarial Networks, Code – Examples of StyleGAN in action: Faces, Anime, Art – Description of the StyleGAN architecture
- Automatic feature engineering using Generative Adversarial Networks with Deeplearning4j & Spark

GAN Use Cases

- Text to Image Generation
- Image to Image Translation
- Increasing Image Resolution
- Predicting Next Video Frame

Notable Papers on GANs

- [Generative Adversarial Nets] [Paper] [Code](Ian Goodfellow's breakthrough paper)

Unclassified Papers & Resources

- GAN Hacks: How to Train a GAN? Tips and tricks to make GANs work
- Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks] [Paper][Code]
- [Adversarial Autoencoders] [Paper][Code]
- [Generating Images with Perceptual Similarity Metrics based on Deep Networks] [Paper]

- [Generating images with recurrent adversarial networks] [Paper] [Code]
- [Generative Visual Manipulation on the Natural Image Manifold] [Paper][Code]
- [Learning What and Where to Draw] [Paper][Code]
- [Adversarial Training for Sketch Retrieval] [Paper]
- [Generative Image Modeling using Style and Structure Adversarial Networks] [Paper][Code]
- [Generative Adversarial Networks as Variational Training of Energy Based Models] [Paper](ICLR 2017)
- [Synthesizing the preferred inputs for neurons in neural networks via deep generator networks] [Paper][Code]
- [SalGAN: Visual Saliency Prediction with Generative Adversarial Networks] [Paper][Code]
- [Adversarial Feature Learning] [Paper]

Generating High-Quality Images

- [Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks] [Paper][Code](Gan with convolutional networks)(ICLR)
- [Generative Adversarial Text to Image Synthesis] [Paper][Code] [Code]
- [Improved Techniques for Training GANs] [Paper][Code] (Goodfellow's paper)
- [Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space] [Paper][Code]
- [StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks] [Paper][Code]
- [Improved Training of Wasserstein GANs] [Paper][Code]
- [Boundary Equibilibrium Generative Adversarial Networks Implementation in Tensorflow] [Paper][Code]
- [Progressive Growing of GANs for Improved Quality, Stability, and Variation] [Paper][Code]

Semi-supervised learning

- [Adversarial Training Methods for Semi-Supervised Text Classification] [Paper][Note](Ian Goodfellow Paper)
- [Improved Techniques for Training GANs] [Paper][Code] (Goodfellow's paper)
- [Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks] [Paper](ICLR)
- [Semi-Supervised QA with Generative Domain-Adaptive Nets] [Paper](ACL 2017)

Ensembles

- [AdaGAN: Boosting Generative Models] [Paper][[Code]]（Google Brain）

Clustering

- [Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks] [Paper](ICLR)

Image blending

- [GP-GAN: Towards Realistic High-Resolution Image Blending] [Paper][Code]

Image Inpainting

- [Semantic Image Inpainting with Perceptual and Contextual Losses] [Paper][Code](CVPR 2017)
- [Context Encoders: Feature Learning by Inpainting] [Paper][Code]
- [Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks] [Paper]
- [Generative face completion] [Paper][Code](CVPR2017)
- [Globally and Locally Consistent Image Completion] [MainPAGE] (SIGGRAPH 2017)

Joint Probability

- [Adversarially Learned Inference][Paper][Code]

Super-Resolution

- [Image super-resolution through deep learning][Code](Just for face dataset)
- [Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network] [Paper][Code]（Using Deep residual network）
- [EnhanceGAN] Docs[[Code]]

De-occlusion

- [Robust LSTM-Autoencoders for Face De-Occlusion in the Wild] [Paper]

Semantic Segmentation

- [Adversarial Deep Structural Networks for Mammographic Mass Segmentation] [Paper][Code]
- [Semantic Segmentation using Adversarial Networks] [Paper]（Soumith's paper）

Object Detection

- [Perceptual generative adversarial networks for small object detection] [Paper](CVPR 2017)

- [A-Fast-RCNN: Hard Positive Generation via Adversary for Object Detection] [Paper][Code](CVPR2017)

RNN-GANs

- [C-RNN-GAN: Continuous recurrent neural networks with adversarial training] [Paper][Code]

Conditional Adversarial Nets

- [Conditional Generative Adversarial Nets] [Paper][Code]
- [InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets] [Paper][Code][Code]
- [Conditional Image Synthesis With Auxiliary Classifier GANs] [Paper][Code](GoogleBrain ICLR 2017)
- [Pixel-Level Domain Transfer] [Paper][Code]
- [Invertible Conditional GANs for image editing] [Paper][Code]
- [Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space] [Paper][Code]
- [StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks] [Paper][Code]
- MaskGAN: Better Text Generation via Filling in the __ Goodfellow et al

Video Prediction & Generation

- [Deep multi-scale video prediction beyond mean square error] [Paper][Code](Yann LeCun's paper)
- [Generating Videos with Scene Dynamics] [Paper][Web][Code]
- [MoCoGAN: Decomposing Motion and Content for Video Generation] [Paper]

Texture Synthesis & Style Transfer

- [Precomputed real-time texture synthesis with markovian generative adversarial networks] [Paper][Code](ECCV 2016)

Image Translation

- [Unsupervised cross-domain image generation] [Paper][Code]
- [Image-to-image translation using conditional adversarial nets] [Paper][Code][Code]
- [Learning to Discover Cross-Domain Relations with Generative Adversarial Networks] [Paper][Code]
- [Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks] [Paper][Code]
- [CoGAN: Coupled Generative Adversarial Networks] [Paper][Code] (NIPS 2016)
- [Unsupervised Image-to-Image Translation with Generative Adversarial Networks] [Paper]

- [Unsupervised Image-to-Image Translation Networks] [Paper]
- [Triangle Generative Adversarial Networks] [Paper]

GAN Theory

- [Energy-based generative adversarial network] [Paper][Code] (Lecun paper)
- [Improved Techniques for Training GANs] [Paper][Code] (Goodfellow's paper)
- [Mode Regularized Generative Adversarial Networks] [Paper] (Yoshua Bengio , ICLR 2017)
- [Improving Generative Adversarial Networks with Denoising Feature Matching] [Paper][Code](Yoshua Bengio , ICLR 2017)
- [Sampling Generative Networks] [Paper][Code]
- [How to train Gans] [Docu]
- [Towards Principled Methods for Training Generative Adversarial Networks] [Paper](ICLR 2017)
- [Unrolled Generative Adversarial Networks] [Paper][Code](ICLR 2017)
- [Least Squares Generative Adversarial Networks] [Paper][Code] (ICCV 2017)
- [Wasserstein GAN] [Paper][Code]
- [Improved Training of Wasserstein GANs] [Paper][Code](The improve of wgan)
- [Towards Principled Methods for Training Generative Adversarial Networks] [Paper]
- [Generalization and Equilibrium in Generative Adversarial Nets] [Paper]（ICML 2017）

3-Dimensional GANs

- [Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling] [Paper][Web][Code](2016 NIPS)
- [Transformation-Grounded Image Generation Network for Novel 3D View Synthesis] [Web](CVPR 2017)

Music

- [MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation using 1D and 2D Conditions] [Paper][HOMEPAGE]

Face Generation & Editing

- [Autoencoding beyond pixels using a learned similarity metric] [Paper][Code][Tensorflow code]
- [Coupled Generative Adversarial Networks] [Paper][Caffe Code] [Tensorflow Code]（NIPS）

- [Invertible Conditional GANs for image editing] [Paper][Code]
- [Learning Residual Images for Face Attribute Manipulation] [Paper] [Code](CVPR 2017)
- [Neural Photo Editing with Introspective Adversarial Networks] [Paper][Code](ICLR 2017)
- [Neural Face Editing with Intrinsic Image Disentangling] [Paper] (CVPR 2017)
- [GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data] [Paper](BMVC 2017)[Code]
- [Beyond Face Rotation: Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis] [Paper](ICCV 2017)

For Discrete Distributions

- [Maximum-Likelihood Augmented Discrete Generative Adversarial Networks] [Paper]
- [Boundary-Seeking Generative Adversarial Networks] [Paper]
- [GANS for Sequences of Discrete Elements with the Gumbel-softmax Distribution] [Paper]

Improving Classification & Recognition

- [Generative OpenMax for Multi-Class Open Set Classification] [Paper](BMVC 2017)
- [Controllable Invariance through Adversarial Feature Learning] [Paper][Code](NIPS 2017)
- [Unlabeled Samples Generated by GAN Improve the Person Re-identification Baseline in vitro] [Paper][Code] (ICCV2017)
- [Learning from Simulated and Unsupervised Images through Adversarial Training] [Paper][Code]（Apple paper, CVPR 2017 Best Paper）

Projects

- [cleverhans] [Code](A library for benchmarking vulnerability to adversarial examples)
- [reset-cppn-gan-tensorflow] [Code](Using Residual Generative Adversarial Networks and Variational Auto-encoder techniques to produce high-resolution images)
- [HyperGAN] [Code](Open source GAN focused on scale and usability)

Tutorials

- [1] Ian Goodfellow's GAN Slides (NIPS Goodfellow Slides)[Chinese Trans]details
- [2] PDF(NIPS Lecun Slides)
- [3] ICCV 2017 Tutorial About GANS