Part 2 - Generative Adversarial Networks

Aykut Erdem

Computer Vision Lab, Hacettepe University
Deep Supervised Learning: A Success Story

• Obtain lots of input-output examples
• Train a deep neural network

Deeplearning

• Achieve superior results
Discriminative vs. Generative Models

\[ p(y|x) \quad \quad p(x|y) \]

Discriminative models

Generative models
Why study deep generative models?

• Go beyond associating inputs to outputs
• Understand high-dimensional, complex probability distributions
• Discover the “true” structure of the data
  • Detect surprising events in the world (*anomaly detection*)
  • Missing Data (*semi-supervised learning*)
  • Generate models for planning (*model-based reinforcement learning*)
Q: What are some recent and potentially upcoming breakthroughs in deep learning?

A: The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks) ... This, and the variations that are now being proposed is the most interesting idea in the last 10 years in ML, in my opinion.
Progress in GANs

Cumulative number of GAN papers by year

Source: https://deephunt.in/the-gan-zoo-79597dc8c347
Generative Modeling

\[ p_{\text{model}} \]

\[ p_{\text{data}} \]
Generative Modeling

Assumptions on $\mathcal{P}$:
- tractable sampling

$p_{\text{model}} \rightarrow p_{\text{data}}$

$p_{\text{data}}$

$p_{\text{model}}$

$p_{\text{data}}$

Training examples
Model samples

Slide adapted from Sebastian Nowozin
Generative Modeling

Assumptions on $\mathcal{P}$:
• tractable sampling
• tractable likelihood function

Slide adapted from Sebastian Nowozin
Three Broad Categories

• Autoregressive Models

• Variational Autoencoders

• Generative Adversarial Networks (GANs)
Autoregressive Models

- Explicitly model conditional probabilities:

\[
p_{\text{model}}(\mathbf{x}) = p_{\text{model}}(x_1) \prod_{i=2}^{n} p_{\text{model}}(x_i | x_1, \ldots, x_{i-1})
\]

Disadvantages:

- Generation can be too costly
- Generation can not be controlled by a latent code

Slide adapted from Ian Goodfellow

BRIEF ARTICLE
THE AUTHOR

Maximum likelihood

\[
\mathbf{x}^\ast = \arg \max_{\mathbf{x}} \mathbb{E}_{\mathbf{x} \sim \text{data}} \log p_{\text{model}}(\mathbf{x} | \mathbf{x})
\]

Fully-visible belief net

\[
p_{\text{model}}(\mathbf{x}) = p_{\text{model}}(x_1) \prod_{i=2}^{n} p_{\text{model}}(x_i | x_1, \ldots, x_{i-1})
\]

Each conditional can be a complicated neural net

Neural Image Model: Pixel RNN

PixelCNN elephants
(van den Ord et al. 2016)
Variational Autoencoder

- Maximizes a variational lower bound on log-likelihood of $\mathbf{x}$

$$\log p(\mathbf{x}) \geq \log p(\mathbf{x}) - D_{KL}(q(z)||p(z | \mathbf{x}))$$

$$= \mathbb{E}_{z \sim q} \log p(\mathbf{x}, z) + H(q)$$

Disadvantages:

- Not asymptotically consistent unless $q$ is perfect
- Tends to produce blurry samples

Face samples for Labeled Faces in the Wild (LFW) (Alec Radford)

Slide adapted from Ian Goodfellow
GANs

Generative Adversarial Networks
Generative Adversarial Networks (GANs)

(Goodfellow et al., 2014)

Advantages:
• A game-theoretic likelihood free model
• Uses a latent code
• No Markov chains needed
• Produces the best samples

Noise (random input)

\[ z \sim \text{Uniform}_{100} \]

think of this as a transformation
Genetive Adversarial Networks (GANs)

(Goodfellow et al., 2014)

- A game between a generator $G_\theta(z)$ and a discriminator $D_\omega(x)$
  - Generator tries to fool discriminator (i.e. generate realistic samples)
  - Discriminator tries to distinguish fake from real samples

\[ \{x_1, \ldots, x_n\} \sim p_{\text{data}} \]
Intuition behind GANs

$D_\omega$ : Discriminator (Art Critic)

$x_{\text{real}}$ $\xrightarrow{\text{green arrow}}$ $G_\theta$ : Generator (Forger)

$x_{\text{fake}}$ $\xrightarrow{\text{red arrow}}$
Intuition behind GAN Training

https://www.youtube.com/watch?v=No26JKQKZNE
Training Procedure

• Use SGD on two minibatches simultaneously:
  - A minibatch of training examples
  - A minibatch of generated samples
GAN Training: Minimax Game (Goodfellow et al., 2014)

\[
\min_{\theta} \max_{\omega} \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log D_\omega (x) \right] + \mathbb{E}_{z \sim p_z} \left[ \log (1 - D_\omega (G_\theta (z))) \right]
\]

\[ J^{(D)} = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \log D(x) - \frac{1}{2} \mathbb{E}_z \log (1 - D (G(z))) \]

\[ J^{(G)} = -\frac{1}{2} \mathbb{E}_z \log D (G(z)) \]

- Equilibrium of the game
- Minimizes the Jensen-Shannon divergence

Real data

Noise vector used to generate data

Cross-entropy loss for binary classification

Generator maximizes the log-probability of the discriminator being mistaken

- Resembles Jensen-Shannon divergence
- Discriminator successfully rejects all generator samples
- Generator maximizes the log-probability of the discriminator
GAN Training: Minimax Game (Goodfellow et al., 2014)

\[
\min_{\theta} \max_{\omega} \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log D_{\omega}(x) \right] + \mathbb{E}_{z \sim p_{z}} \left[ \log (1 - D_{\omega}(G_{\theta}(z))) \right]
\]

- Equilibrium of the game
- Minimizes the Jensen-Shannon divergence

**Important question is**

"Does this converge??"

• Equilibrium of the game
• Minimizes the Jensen-Shannon divergence

- \(J(D) = -\frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} [\log D_{\omega}(x)]\)
- \(J(G) = -\frac{1}{2} \mathbb{E}_{z \sim p_{z}} [\log (1 - D_{\omega}(G_{\theta}(z)))\)

- Cross-entropy loss for binary classification
- Noise vector used to model probability of the discriminator being mistaken
- Resembles Jensen-Shannon divergence
- Equilibrium no longer describable with a single loss
Training Procedure

 Generating 1D points

 (Goodfellow et al., 2014)

 Generating images

Source: Alec Radford

Source: OpenAI blog
Results

(Goodfellow et al., 2014)

- The generator uses a mixture of rectifier linear activations and/or sigmoid activations.
- The discriminator net used maxout activations.

![MNIST samples](image)

![TFD samples](image)

![CIFAR10 samples](image)

(CIFAR10 samples (fully-connected model))

(CIFAR10 samples (convolutional discriminator, deconvolutional generator))
Laplacian GANs (LAPGAN) (Denton et al., 2015)

**Idea:** Combine GAN with a multi-scale image representation (Laplacian pyramid)

- Idea: Combine GAN with a multi-scale image representation (Laplacian pyramid)
LAPGAN for LSUN Towers

64×64 pixels
~700K images

(Denton et al., 2015)
LAPGAN for LSUN Bedrooms

64×64 pixels

~3M images (Denton et al., 2015)
Deep Convolutional GANs (DCGAN)

• **Idea:** Tricks to make GAN training more stable

- No fully connected layers
- Batch Normalization (Ioffe and Szegedy, 2015)
- Leaky Rectifier in $D$
- Use Adam (Kingma and Ba, 2015)
- Tweak Adam hyperparameters a bit ($lr=0.0002$, $b1=0.5$)

(Radford et al., 2015)
DCGAN for LSUN Bedrooms 64×64 pixels ~3M images (Radford et al., 2015)
Walking over the latent space (Radford et al., 2015)

- Interpolation suggests non-overfitting behavior
Walking over the latent space  (Radford et al., 2015)
Vector Space Arithmetic

(Radford et al., 2015)
Vector Space Arithmetic

(smiling woman) - (neutral woman) + (neutral man) = (smiling man)

(Radford et al., 2015)
Subclasses of GANs

Vanilla GAN
(Goodfellow, et al., 2014)

Conditional GAN
(Mirza & Osindero, 2014)

Bidirectional GAN
(Donahue, et al., 2016; Dumoulin, et al., 2016)

Semi-Supervised GAN
(Ono, 2016; Salimans, et al., 2016)

InfoGAN
(Chen, et al., 2016)

Auxiliary Classifier GAN
(Ono, et al., 2016)

Image: Christopher Olah
Vanilla GAN (Goodfellow et al., 2014)

DCGAN (Radford et al., 2015)
Conditional GAN (Mirza and Osindero, 2014)

- Add conditional variables $y$ into $G$ and $D$

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

Fig 1 illustrates the structure of a simple conditional adversarial net.
Conditional GAN (Mirza and Osindero, 2014)

- Add conditional variables $y$ into $G$ and $D$

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

Fig 1 illustrates the structure of a simple conditional adversarial net.

4 Experimental Results

4.1 Unimodal

We trained a conditional adversarial net on MNIST images conditioned on their class labels, encoded as one-hot vectors. In the generator net, a noise prior $z$ with dimensionality 100 was drawn from a uniform distribution within the unit hypercube. Both $z$ and $y$ are mapped to hidden layers with Rectified Linear Unit (ReLU) activation [4, 11], with layer sizes 200 and 1000 respectively, before both being mapped to second, combined hidden ReLu layer of dimensionality 1200. We then have a final sigmoid unit layer as our output for generating the 784-dimensional MNIST samples.

For now we simply have the conditioning input and prior noise as inputs to a single hidden layer of a MLP, but one could imagine using higher order interactions allowing for complex generation mechanisms that would be extremely difficult to work with in a traditional generative framework.

More on Conditional GANs in Part 3 (Levent)
Auxiliary Classifier GAN (Odena et al., 2016)

• Every generated sample has a corresponding class label

\[
L_S = E[\log P(S = real \mid X_{real})] + E[\log P(S = fake \mid X_{fake})]
\]

\[
L_C = E[\log P(C = c \mid X_{real})] + E[\log P(C = c \mid X_{fake})]
\]

• \(D\) is trained to maximize \(L_S + L_C\)

• \(G\) is trained to maximize \(L_C - L_S\)

• Learns a representation for \(z\) that is independent of class label
Auxiliary Classifier GAN (Odena et al., 2016)

128×128 resolution samples from 5 classes taken from an AC-GAN trained on the ImageNet

monarch butterfly
goldfinch
daisy
redshank
grey whale
Bidirectional GAN (Donahue et al., 2016; Dumoulin et al., 2016)

- Jointly learns a generator network and an inference network using an adversarial process.

\[
\min_G \max_D V(D, G) = \mathbb{E}_{q(x)}[\log(D(x, G_z(x)))] + \mathbb{E}_{p(z)}[\log(1 - D(G_x(z), z))]
\]

\[
= \int \int q(x)q(z | x) \log(D(x, z)) dx dz
\]

\[
+ \int \int p(z)p(x | z) \log(1 - D(x, z)) dx dz.
\]

CelebA reconstructions

SVNH reconstructions
Bidirectional GAN (Donahue et al., 2016; Dumoulin et al., 2016)

LSUN bedrooms

Tiny ImageNet
Applications of GANs
Semi-supervised Classification

(Salimans et al., 2016; Dumoulin et al., 2016)

Figure 6: Latent space interpolations on the CelebA validation set. Left and right columns correspond to the original pairs $x_1$ and $x_2$, and the columns in between correspond to the decoding of latent representations interpolated linearly from $z_1$ to $z_2$. Unlike other adversarial approaches like DCGAN (Radford et al., 2015), ALI allows one to interpolate between actual data points. Using ALI's inference network as opposed to the discriminator to extract features, we achieve a misclassification rate that is roughly $3 \pm 0.5\%$ lower than reported in Radford et al. (2015) (Table 1), which suggests that ALI's inference mechanism is beneficial to the semi-supervised learning task.

We then investigate ALI's performance when label information is taken into account during training. We adapt the discriminative model proposed in Salimans et al. (2016). The discriminator takes $x$ and $z$ as input and outputs a distribution over $K + 1$ classes, where $K$ is the number of categories. When label information is available for $q(x, z)$ samples, the discriminator is expected to predict the label. When no label information is available, the discriminator is expected to predict $K + 1$ for $p(x, z)$ samples and $k \{1, \ldots, K\}$ for $q(x, z)$ samples.

Interestingly, Salimans et al. (2016) found that they required an alternative training strategy for the generator where it tries to match first-order statistics in the discriminator's intermediate activations with respect to the data distribution (they refer to this as feature matching). We found that ALI did not require feature matching to obtain comparable results. We achieve results competitive with the state-of-the-art, as shown in Tables 1 and 2. Table 2 shows that ALI offers a modest improvement over Salimans et al. (2016), more specifically for 1000 and 2000 labeled examples.

### Table 1: SVHN test set misclassification rate

<table>
<thead>
<tr>
<th>Model</th>
<th>Misclassification rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAE (M1 + M2) (Kingma et al., 2014)</td>
<td>36.02</td>
</tr>
<tr>
<td>SWWAE with dropout (Zhao et al., 2015)</td>
<td>23.56</td>
</tr>
<tr>
<td>DCGAN + L2-SVM (Radford et al., 2015)</td>
<td>22.18</td>
</tr>
<tr>
<td>SDGM (Maaløe et al., 2016)</td>
<td>16.61</td>
</tr>
<tr>
<td>GAN (feature matching) (Salimans et al., 2016)</td>
<td>$8.11 \pm 1.3$</td>
</tr>
<tr>
<td>ALI (ours, L2-SVM)</td>
<td>$19.14 \pm 0.50$</td>
</tr>
<tr>
<td>ALI (ours, no feature matching)</td>
<td>$7.42 \pm 0.65$</td>
</tr>
</tbody>
</table>

### Table 2: CIFAR10 test set misclassification rate for semi-supervised learning using different numbers of trained labeled examples. For ALI, error bars correspond to 3 times the standard deviation.

<table>
<thead>
<tr>
<th>Number of labeled examples</th>
<th>Model</th>
<th>Misclassification rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Ladder network (Rasmus et al., 2015)</td>
<td>$20 \pm 0.40$</td>
</tr>
<tr>
<td></td>
<td>CatGAN (Springenberg, 2015)</td>
<td>$19 \pm 0.58$</td>
</tr>
</tbody>
</table>
|                            | GAN (feature matching) (Salimans et al., 2016) | $21 \pm 0.83 \pm 2 \pm 0.01 19 
|                            | ALI (ours, no feature matching)                 | $19 \pm 0.98 \pm 0.89 19 
|                            | ALI (ours, L2-SVM)                              | $19 \pm 0.62 \pm 0.44 17 
|                            | ALI (ours, L2-SVM)                              | $17 \pm 0.99 \pm 0.62 17 
| 4000                       | Ladder network (Rasmus et al., 2015)            | $20 \pm 0.40$          |
|                            | CatGAN (Springenberg, 2015)                     | $19 \pm 0.58$          |
|                            | GAN (feature matching) (Salimans et al., 2016) | $21 \pm 0.83 \pm 2 \pm 0.01 19 
|                            | ALI (ours, no feature matching)                 | $19 \pm 0.98 \pm 0.89 19 
|                            | ALI (ours, L2-SVM)                              | $19 \pm 0.62 \pm 0.44 17 
| 8000                       | Ladder network (Rasmus et al., 2015)            | $20 \pm 0.40$          |
|                            | CatGAN (Springenberg, 2015)                     | $19 \pm 0.58$          |
|                            | GAN (feature matching) (Salimans et al., 2016) | $21 \pm 0.83 \pm 2 \pm 0.01 19 
|                            | ALI (ours, no feature matching)                 | $19 \pm 0.98 \pm 0.89 19 
|                            | ALI (ours, L2-SVM)                              | $19 \pm 0.62 \pm 0.44 17 

SVNH
Class-specific Image Generation (Nguyen et al., 2016)

- Generates 227x227 realistic images from all ImageNet classes
- Combines adversarial training, moment matching denoising autoencoders, and Langevin sampling
Video Generation (Vondrick et al., 2016)

Beach

Golf

Train Station
Generative Shape Modeling (Wu et al., 2016)

We have discussed how to generate 3D objects by sampling a latent vector $z_0$ in 3D voxel space. The discriminator $D$ observes this helps to stabilize the training and to produce better results. We set the learning rate of the generator to keep the training of both networks in pace, we employ an adaptive training strategy: for each batch, the discriminator only gets updated if its accuracy in the last batch is not higher than 80%. We use ADAM instead of ReLU layers. There are no pooling or linear layers in our network. More details can be found in the supplementary material.

As proposed in Goodfellow et al. [2014], we use binary cross entropy as the classification loss, and present our overall adversarial loss function as $\mathcal{L}^{\text{3D-GAN}} = \mathbb{E}_{x \sim p_{\text{data}}}(\log D(x)) + \mathbb{E}_{z \sim p_{\text{z}}}(\log (1 - D(G(z))))$, where $G(z)$ is a randomly sampled noise vector from a probabilistic latent space, to a generator and a discriminator, where the discriminator tries to classify real objects and objects synthesized by the generator, and the generator attempts to confuse the discriminator. In our 3D modeling 2D images, we discuss the use of an adversarial component in modeling 3D objects.

3D Generative Adversarial Network (3D-GAN)

A straightforward training procedure is to update both the generator and the discriminator in every batch. However, the discriminator usually learns much faster than the generator, possibly because generating objects in a 3D voxel space is more difficult than differentiating between real and synthetic objects. Therefore, for the generator to extract signals for improvement from a discriminator that is way ahead, as all examples it generated would be correctly identified as synthetic with high confidence. Therefore, we set the optimizer of the generator to use Leaky ReLU with $\alpha = 0.2$ and mapping it to the latent space. In practice, it would also be helpful to infer these latent vectors from observations. For optimization, with $G(z)$ we observe this helps to stabilize the training and to produce better results. We set the learning rate of the generator to keep the training of both networks in pace, we employ an adaptive training strategy: for each batch, the discriminator only gets updated if its accuracy in the last batch is not higher than 80%.

In this work, each dimension of $z$ is real or synthetic. For each dimension of $z$, it is randomly sampled from a probabilistic latent space, to a generator and a discriminator, where the discriminator tries to classify real objects and objects synthesized by the generator, and the generator attempts to confuse the discriminator. In our 3D modeling 2D images, we discuss the use of an adversarial component in modeling 3D objects.

For the generator, except that it uses Leaky ReLU with $\alpha = 0.2$, we add in between and a Sigmoid layer at the end. The discriminator basically mirrors the generator, except that it uses Leaky ReLU with $\alpha = 0.2$ and strides $4 \times 4 \times 4$.
Text-to-Image Synthesis (Zhang et al., 2016)

The small bird has a red head with feathers that fade from red to gray from head to tail.

The petals of this flower are white with a large stigma.

A unique yellow flower with no visible pistils protruding from the center.

This flower is pink and yellow in color, with petals that are oddly shaped.

This is a light colored flower with many different petals on a green stem.

This flower is yellow and green in color, with petals that are ruffled.

The flower have large petals that are pink with yellow on some of the petals.

A flower that has white petals with some tones of yellow and green filaments.
Single Image Super-Resolution (Ledig et al., 2016)

- Combine content loss with adversarial loss
Image Inpainting (Pathak et al., 2016)
Unsupervised Domain Adaptation (Bousmalis et al., 2016)

Image examples from the Linemod dataset

RGDB image samples (conditioned on a synthetic image)
Image Editing (Karacan et al., 2016)

“Maybe in our world lives a happy little tree over there.”
— Bob Ross
How to Evaluate GANs?
In this task, we present you computer generated pictures of outdoor scenes generated by different computer programs. Your task is to compare them and determine which is more realistic and natural looking. See the below table for some examples.

### Steps

1. Analyze both images and consider their features carefully
2. Determine which computer generated image (image A or image B) is more realistic than the other.

Which image seems more real?

- [ ] A
- [ ] B

From (Karacan et al., 2016)
Evaluating Quality

- Hard to tell if progress is being made by looking at losses
- Inception Score (Salimans et al., 2016)
- Inception Accuracy (Odena et al., 2016)
  - Report the fraction of the samples for which the Inception network assigned the correct label
Measuring Diversity (Odena et al., 2016)

- **MS-SSIM scores** [between randomly chosen pairs of images within a given class]

<table>
<thead>
<tr>
<th>Object</th>
<th>MS-SSIM Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hot dog</td>
<td>0.11</td>
</tr>
<tr>
<td>Promontory</td>
<td>0.29</td>
</tr>
<tr>
<td>Green apple</td>
<td>0.41</td>
</tr>
<tr>
<td>Artichoke</td>
<td>0.90</td>
</tr>
</tbody>
</table>

**Figure 4:** Examples of different MS-SSIM scores. The top and bottom rows contain AC-GAN samples and training data, respectively.

**Figure 5:** (Left) Comparison of the mean MS-SSIM scores between pairs of images within a given class for ImageNet training data and samples from the GAN (blue line is equality). The horizontal red line marks the maximum MS-SSIM value across all ImageNet classes. Each point is an individual class. The mean standard deviation of scores across the training data and the samples was 0.06 and 0.08 respectively. Scores below the red line (84.7% of classes) arise from classes where GAN training largely succeeded. (Right) Intra-class MS-SSIM for selected ImageNet classes throughout a training run. Classes that successfully train tend to have decreasing mean MS-SSIM scores, to a point.

SSIM values between 0.25 to 1 contain Inception accuracies ≥1%. These results suggest that GANs that drop modes are most likely to produce low quality images. Conversely, 78% of classes with high diversity (MS-SSIM < 0.25) have Inception accuracies that exceed 1%. In comparison, the Inception-v3 model achieves 78.8% accuracy on average across all 1000 classes (Szegedy et al., 2015). A fraction of the classes AC-GAN samples reach this level of accuracy. This indicates opportunity for future image synthesis models.

4.4 Comparison to Previous Results

Previous quantitative results for image synthesis models trained on ImageNet are reported in terms of log-likelihood (van den Oord et al., 2016a;b). Log-likelihood is a coarse and potentially inaccurate measure of sample quality (Theis et al., 2015). Additionally, log-likelihood is intractable to compute for GANs. Instead we compare with previous state-of-the-art results on CIFAR-10 using a lower spatial resolution (32×32). Following the procedure in Salimans et al. (2016), we compute...
Searching for Overfitting

• Nearest Neighbor Analysis (Odena et al., 2016)

Table 1: SVHN test set misclassification rate for semi-supervised learning using different numbers of trained labeled examples. For ALI, error bars correspond to 3 times the standard deviation.

Table 2: CIFAR10 test set misclassification rate for semi-supervised learning using different numbers of trained labeled examples. For ALI, error bars correspond to 3 times the standard deviation.

• Latent Space Interpolations

Image: (Dumolil et al., 2016)
Limitations
Cherry-Picked Results

Slide credit: Ian Goodfellow
Problems with Counting
Problems with Perspective

Slide credit: Ian Goodfellow
Problems with Perspective

This one was real
Problems with Global Structure
**Mode Collapse** (Metz et al., 2016)

\[
\min_G \max_D V(G, D) \neq \max_D \min_G V(G, D)
\]

- \(D\) in inner loop: convergence to correct distribution
- \(G\) in inner loop: place all mass on most likely point

![Target](image)

**Step**
- **0**
- **5k**
- **10k**
- **15k**
- **20k**
- **25k**
Mode collapse causes low output diversity

(Reed et al. 2016)
Non-convergence

• Optimization algorithms often approach a saddle point or local minimum rather than a global minimum

• Game solving algorithms may not approach an equilibrium at all
Wasserstein GAN (Arjovsky et al., 2016)

- Objective based on Earth-Mover or Wassertein distance:
  \[
  \min_{\theta} \max_{\omega} \mathbb{E}_{x \sim p_{\text{data}}} [D_\omega(x)] - \mathbb{E}_{z \sim p_z} [D_\omega(G_\theta(z))]
  \]

- Provides nice gradients over real and fake samples
**Wasserstein GAN** (Arjovsky et al., 2016)

- Wasserstein loss seems to correlate well with image quality.

---

![Figure 3](image-url)
WGAN with gradient penalty (Gulraani et al., 2017)

\[
L = \mathbb{E}_{\hat{x} \sim P_g} [D(\hat{x})] - \mathbb{E}_{x \sim P_r} [D(x)] + \lambda \mathbb{E}_{\hat{x} \sim P_{\hat{x}}} [(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2]
\]

Original critic loss  
Our gradient penalty

- Faster convergence and higher-quality samples than WGAN with weight clipping
- Train a wide variety of GAN architectures with almost no hyperparameter tuning, including discrete models

Samples from a character-level GAN language model on Google Billion Word

<table>
<thead>
<tr>
<th>Standard GAN objective</th>
<th>Solice Norkedin pring in since ThiS record ( 31. ) UBS ) and Ch</th>
</tr>
</thead>
<tbody>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>These leaded as most-worsd p2 a0</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>The time I paidDa South Cubry i</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>Dour Fraps higs it was these del</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>This year out howned allowed lo</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>Kaulna Seto consficates to repor</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>A can teaal , he was schoon news</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>In th 200. Pesish picriers rega</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>Konny Panice rimirmer the teami</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>The new centcut cut Denester of</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>The near , had been one injustie</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>The incestion to week to shorted</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>The company the high product of</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>20 - The time of accomplate , wh</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>John WUnderenson seqivic spends</td>
</tr>
<tr>
<td>ddddddddddddddddddddddddddddddddd</td>
<td>A ceetens in indestredly the Wat</td>
</tr>
</tbody>
</table>
Boundary Equilibrium GAN (BEGAN) (Berthelot et al., 2017)

- A loss derived from the Wasserstein distance for training auto-encoder based GANs
  \[ \mathcal{L}(v) = |v - D(v)|^\eta \] where \( D : \mathbb{R}^{N_x} \mapsto \mathbb{R}^{N_x} \) is the autoencoder function, \( \eta \in \{1, 2\} \) is the target norm, and \( v \in \mathbb{R}^{N_x} \) is a sample of dimension \( N_x \).

- Wasserstein distance btw. the reconstruction losses of real and generated data

- Convergence measure:
  \[ \mathcal{M}_{global} = \mathcal{L}(x) + |\gamma \mathcal{L}(x) - \mathcal{L}(G(z_G))| \]

- Objective:
  \[
  \begin{align*}
  \mathcal{L}_D &= \mathcal{L}(x) - k_t \mathcal{L}(G(z_D)) \\
  \mathcal{L}_G &= \mathcal{L}(G(z_G)) \\
  k_{t+1} &= k_t + \lambda_k (\gamma \mathcal{L}(x) - \mathcal{L}(G(z_G)))
  \end{align*}
  \]
  for \( \theta_D \) for \( \theta_G \) for each training step \( t \).
Sample diversity, while not perfect, is convincing; the generated images look relatively close to the real ones. The interpolations show good continuity. On the first row, the hair transitions in a natural way and intermediate hairstyles are believable, showing good generalization. It is also worth noting that some features are not represented such as the cigarette in the left image. The second and last rows show simple rotations. While the rotations are smooth, we can see that profile pictures are not captured as well as camera facing ones. We assume this is due to profiles being less common in our dataset. Finally the mirror example demonstrates separation between identity and rotation. A surprisingly realistic camera-facing image is derived from a single profile image.

4.4 Convergence measure and image quality

The convergence measure $M_{global}$ was conjectured earlier to measure the convergence of the BEGAN model. As can be seen in figure 5 this measure correlates well with image fidelity. We can also...