# Generative Adversarial Networks for Unsupervised Fault Detection

2018 European Control Conference (ECC)

by P. Spyridon and Y. S. Boutalis

---

Lee Rippon

The University of British Columbia
Chemical and Biological Engineering
Pulp and Paper Center
DAIS Lab

October 30, 2019

## Outline

- **Application:** simple fault detection, i.e., identifying deviations from normal operation.

- **Application:** simple fault detection, i.e., identifying deviations from normal operation.
- **Process:** Tennessee Eastman Process. Industrial benchmark data-set for fault detection, diagnosis, isolation and alarm management.

- **Application:** simple fault detection, i.e., identifying deviations from normal operation.
- **Process:** Tennessee Eastman Process. Industrial benchmark data-set for fault detection, diagnosis, isolation and alarm management.
- **Flavor:**

- **Application:** simple fault detection, i.e., identifying deviations from normal operation.
- **Process:** Tennessee Eastman Process. Industrial benchmark data-set for fault detection, diagnosis, isolation and alarm management.
- **Flavor:**
  - Pattern recognition and classification.

- **Application:** simple fault detection, i.e., identifying deviations from normal operation.
- **Process:** Tennessee Eastman Process. Industrial benchmark data-set for fault detection, diagnosis, isolation and alarm management.
- **Flavor:**
    - Pattern recognition and classification.
    - One-class classification (i.e., anomaly detection).

- **Application:** simple fault detection, i.e., identifying deviations from normal operation.
- **Process:** Tennessee Eastman Process. Industrial benchmark data-set for fault detection, diagnosis, isolation and alarm management.
- **Flavor:**
  - Pattern recognition and classification.
  - One-class classification (i.e., anomaly detection).
  - Generative methods.

- Fault detection methods are divided into the following three categories:

- Fault detection methods are divided into the following three categories:

  1. **Model based methods -** calculating residuals $\epsilon = |y - \hat{y}|$.

- Fault detection methods are divided into the following three categories:

  1. **Model based methods -** calculating residuals $\epsilon = |y - \hat{y}|$.
  2. **Signal processing based methods -** monitoring statistical operations of the system signals.

- Fault detection methods are divided into the following three categories:

    1. **Model based methods** - calculating residuals $\epsilon = |y - \hat{y}|$.
    2. **Signal processing based methods** - monitoring statistical operations of the system signals.
    3. **Pattern recognition or classification methods** - finding a boundary condition between classes.

- **One-class classification (OCC):** training data includes only one class of data (normal operation). Detect deviations from normal operation.

- **One-class classification (OCC):** training data includes only one class of data (normal operation). Detect deviations from normal operation.
    - **Motivation:** abnormal scenarios are disastrous and rare so the set of faulty data for training is limited.

- **One-class classification (OCC):** training data includes only one class of data (normal operation). Detect deviations from normal operation.
    - **Motivation:** abnormal scenarios are disastrous and rare so the set of faulty data for training is limited.
    - **OCC algorithms:** many proposed in literature using neural networks, nearest neighbors, decision trees and bayesian classifiers.

- **One-class classification (OCC):** training data includes only one class of data (normal operation). Detect deviations from normal operation.
  - **Motivation:** abnormal scenarios are disastrous and rare so the set of faulty data for training is limited.
  - **OCC algorithms:** many proposed in literature using neural networks, nearest neighbors, decision trees and bayesian classifiers.
  - **Benchmark:** One-class support vector machines (OCSVM). Popular due to ability to handle non-linear decision boundaries.

- Typical segue into **deep learning** (DL) methods:

- Typical segue into **deep learning** (DL) methods:
  - Extract features at various levels of representation, achieve a hierarchical representation for learning.

- Typical segue into **deep learning** (DL) methods:
  - Extract features at various levels of representation, achieve a hierarchical representation for learning.
  - Immense success and popularity in computer vision, natural language processing and audio recognition.

- Typical segue into **deep learning** (DL) methods:
  - Extract features at various levels of representation, achieve a hierarchical representation for learning.
  - Immense success and popularity in computer vision, natural language processing and audio recognition.
  - Application in fault and anomaly detection such as acoustic novelty detection using auto-encoders.

- Typical segue into **deep learning** (DL) methods:
  - Extract features at various levels of representation, achieve a hierarchical representation for learning.
  - Immense success and popularity in computer vision, natural language processing and audio recognition.
  - Application in fault and anomaly detection such as acoustic novelty detection using auto-encoders.
- **Generative models** - estimate the probability distribution function of the training data-set. Popular methods include:

- Typical segue into **deep learning** (DL) methods:
  - Extract features at various levels of representation, achieve a hierarchical representation for learning.
  - Immense success and popularity in computer vision, natural language processing and audio recognition.
  - Application in fault and anomaly detection such as acoustic novelty detection using auto-encoders.
- **Generative models** - estimate the probability distribution function of the training data-set. Popular methods include:
  - Autoregressive models.
  - Variational Auto-Encoders (VAEs).
  - Generative Adversarial Networks (GANs).

- **Generator** attempts to create realistic samples that fool the discriminator. **Discriminator** is trying to classify if a sample is real or fake. **Non-cooperative game theory.**

- **Generator** attempts to create realistic samples that fool the discriminator. **Discriminator** is trying to classify if a sample is real or fake. **Non-cooperative game theory.**
- Two stochastic gradient descent (SGD) algorithms run simultaneously to update parameters of the generator and discriminator.

- **Generator** attempts to create realistic samples that fool the discriminator. **Discriminator** is trying to classify if a sample is real or fake. **Non-cooperative game theory.**
- Two stochastic gradient descent (SGD) algorithms run simultaneously to update parameters of the generator and discriminator.

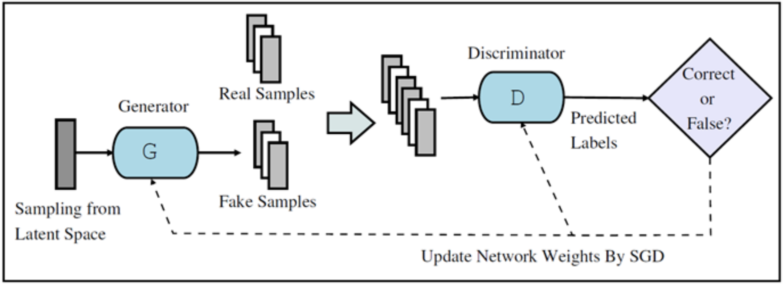


Figure 1: Architecture of Generative Adversarial Network.

- The discriminator output, $D(x)$, is the estimated probability that $x$ comes from the training data rather than the generator.

- The discriminator output, $D(x)$, is the estimated probability that $x$ comes from the training data rather than the generator.

- The discriminator cost function, $J^D$, is the cross-entropy between the discriminator output and the actual labels:

$$
\begin{aligned}
J^D(\theta^D, \theta^G) = &- \frac{1}{2}\mathbb{E}_{x \sim p_{\text{data}}} \log(D(x)) \\
&- \frac{1}{2}\mathbb{E}_{z \sim P(z)} \log(1 - D(G(z))).
\end{aligned} \tag{1}
$$

- The discriminator output, $D(x)$, is the estimated probability that $x$ comes from the training data rather than the generator.

- The discriminator cost function, $J^D$, is the cross-entropy between the discriminator output and the actual labels:

$$\begin{aligned} J^D(\theta^D, \theta^G) = & -\frac{1}{2}\mathbb{E}_{x\sim p_{\text{data}}} \log(D(x)) \\ & -\frac{1}{2}\mathbb{E}_{z\sim P(z)} \log(1 - D(G(z))). \end{aligned} \tag{1}$$

- The generator cost function ($J^G$) attempts to maximize the log probability of false discriminator decision, i.e.,

$$J^G(\theta^G, \theta^D) = -\frac{1}{2}\mathbb{E}_{z\sim P(z)} \log(D(G(z))). \tag{2}$$

- Disadvantage of GAN architecture is training instability.

- Disadvantage of GAN architecture is training instability.
- Solution is not a local minimum of the cost function, instead it is the **Nash equilibrium** of the game, i.e., where each network has no benefit of changing strategy.

- Disadvantage of GAN architecture is training instability.
- Solution is not a local minimum of the cost function, instead it is the **Nash equilibrium** of the game, i.e., where each network has no benefit of changing strategy.
- **Mode collapse** is a common cause of instability.

- Disadvantage of GAN architecture is training instability.
- Solution is not a local minimum of the cost function, instead it is the **Nash equilibrium** of the game, i.e., where each network has no benefit of changing strategy.
- **Mode collapse** is a common cause of instability.

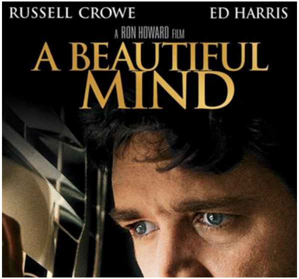


Figure 2: John Forbes Nash Jr. played by Russel Crowe in *A Beautiful Mind*, a biographical drama film from 2001).

- Several approaches to enhance GAN stability:

- Several approaches to enhance GAN stability:
  - **Feature matching:** altering generator objective by using intermediate discriminator output.

- Several approaches to enhance GAN stability:
  - **Feature matching:** altering generator objective by using intermediate discriminator output.
  - Addition of **instant noise** to both types of samples before entering discriminator.

- Several approaches to enhance GAN stability:
  - **Feature matching:** altering generator objective by using intermediate discriminator output.
  - Addition of **instant noise** to both types of samples before entering discriminator.
  - Using **dropout** to increase regularization of the generator and avoid over-fitting.

- **Quantitative evaluation** is an open problem of the GAN framework. Approaches include:

- **Quantitative evaluation** is an open problem of the GAN framework. Approaches include:
  - Nearest neighbor evaluation from generated samples.

- **Quantitative evaluation** is an open problem of the GAN framework. Approaches include:
  - Nearest neighbor evaluation from generated samples.
  - Classification performance of the generator.

- **Quantitative evaluation** is an open problem of the GAN framework. Approaches include:
  - Nearest neighbor evaluation from generated samples.
  - Classification performance of the generator.
  - Comparing two GAN models through a competitive game.

- **Quantitative evaluation** is an open problem of the GAN framework. Approaches include:
  - Nearest neighbor evaluation from generated samples.
  - Classification performance of the generator.
  - Comparing two GAN models through a competitive game.
  - Evaluating the reconstruction cost of an autoencoder.

- **Autoencoders (AEs):** A feedforward neural network that maps inputs to a hidden latent representation (**encoding**) and then mapping the latent representation back to the original space (**decoding**).

- **Autoencoders (AEs):** A feedforward neural network that maps inputs to a hidden latent representation (**encoding**) and then mapping the latent representation back to the original space (**decoding**).

- Try to minimize the **reconstruction cost** of the input.

- **Autoencoders (AEs):** A feedforward neural network that maps inputs to a hidden latent representation (**encoding**) and then mapping the latent representation back to the original space (**decoding**).

- Try to minimize the **reconstruction cost** of the input.

- **Evaluation approach:** train AE using generator samples and evaluate reconstruction cost $C^{\text{fake}}$. Input the original data and evaluate the reconstruction cost $C^{\text{real}}$.

- **Autoencoders (AEs):** A feedforward neural network that maps inputs to a hidden latent representation (**encoding**) and then mapping the latent representation back to the original space (**decoding**).

- Try to minimize the **reconstruction cost** of the input.

- **Evaluation approach:** train AE using generator samples and evaluate reconstruction cost $C^{\text{fake}}$. Input the original data and evaluate the reconstruction cost $C^{\text{real}}$.

- Maximize the evaluation grade $g^{\text{eval}}$

$$g^{\text{eval}} = \frac{C^{\text{fake}}}{C^{\text{real}}}. \tag{3}$$

- For a set of GAN models, return the model with the highest $g^{\text{eval}}$.

- For a set of GAN models, return the model with the highest $g^{\text{eval}}$.

---

**Algorithm 1** Calculate the evaluation grade $g^{eval}$ of GAN model based on AE

---

1. Train the GAN model using the training dataset of real samples
2. Generate a batch of fake samples with equal size as the batch of real ones
3. Train an AE with training dataset the batch of fake samples and estimate $C^{fake}$
4. Calculate the reconstruction cost of AE for the set of real samples $C^{real}$, without training
5. Return the ratio $\frac{C^{fake}}{C^{real}}$

---

- Generator and discriminator **hyper-parameters** are key considerations to maintain Nash equilibrium.

- Generator and discriminator **hyper-parameters** are key considerations to maintain Nash equilibrium.
- A grid search algorithm is used to select the model hyper-parameters.

- Generator and discriminator **hyper-parameters** are key considerations to maintain Nash equilibrium.
- A grid search algorithm is used to select the model hyper-parameters.
- Final model selection decision is taken by calculating the $g$-mean accuracy, i.e., $g^{\text{mean}} = \sqrt{a^+ a^-}$ where $a^+$ and $a^-$ is the classification accuracy on normal and abnormal data, respectively.

**Algorithm 2** Model Selection of GAN for unsupervised classification

1. Construct the grid of hyper-parameters
2. Train $N$ GAN models, where each model corresponds to a combination of hyper-parameters ($N$ is the number of hyper-parameter combinations)
3. For each trained model $i$, calculate the evaluation grade $g_i^{eval}$ using algorithmic procedure 1
4. Calculate the normalized evaluation grade of each model with the equation $g_i^{neval} = \frac{g_i^{eval}}{\max_i g_i^{eval}}$
5. Find the accuracy of the Discriminator on real samples $a_i^+$ for each model $i$.
6. Find the accuracy of the Discriminator on fake samples $a_i^-$ for each model $i$.
7. The accuracy on fake samples for each model is updated as $a_i^- = a_i^- * g_i^{neval}$
8. The g-mean of each model calculated by the equation $g_i^{mean} = \sqrt{a_i^+ a_i^-}$
9. Finally, select the model $j$ where $j = \text{argmin}_i\, g_i^{mean}$

- The TE process data-set contains 52 process variables and 21 faulty scenarios.

- The TE process data-set contains 52 process variables and 21 faulty scenarios.
- The normal class training set is 480 samples and the testing set is 960 samples. In this paper these are combined into a normal training set of length 1440.

- The TE process data-set contains 52 process variables and 21 faulty scenarios.
- The normal class training set is 480 samples and the testing set is 960 samples. In this paper these are combined into a normal training set of length 1440.
- Each faulty class has a testing set of 960 samples where the first 160 samples (8 hours) are normal operation.

- Both the generator and discriminator networks have the follow characteristics:

- Both the generator and discriminator networks have the follow characteristics:
    - 5 hidden layers (fixed number of nodes per layer)

- Both the generator and discriminator networks have the follow characteristics:
  - 5 hidden layers (fixed number of nodes per layer)
  - Leaky rectified linear units (ReLU) activation functions.

- Both the generator and discriminator networks have the follow characteristics:
  - 5 hidden layers (fixed number of nodes per layer)
  - Leaky rectified linear units (ReLU) activation functions.
  - Trained with the Adam optimizer.

- Both the generator and discriminator networks have the follow characteristics:
  - 5 hidden layers (fixed number of nodes per layer)
  - Leaky rectified linear units (ReLU) activation functions.
  - Trained with the Adam optimizer.
- The output layer for the generator has linear mappings whereas the output layer of the discriminator uses the softmax function.

- Both the generator and discriminator networks have the follow characteristics:
  - 5 hidden layers (fixed number of nodes per layer)
  - Leaky rectified linear units (ReLU) activation functions.
  - Trained with the Adam optimizer.
- The output layer for the generator has linear mappings whereas the output layer of the discriminator uses the softmax function.
- Additive noise, dropout and feature mapping are used to boost the stability of the generator.

TABLE I

EVALUATION OF GAN MODELS WITH SEARCH ALGORITHM

| Model | *Generator Nodes* | *Discriminator Nodes* | *Sampling dimension* | *Evaluating g-mean* | *Testing g-mean* |
|---|---|---|---|---|---|
| 1 | 400 | 600 | 500 | .882 | 73.76% |
| 2 | **400** | **600** | **1000** | **.953** | **75.04%** |
| 3 | 400 | 800 | 500 | .806 | 74.66% |
| 4 | 400 | 800 | 1000 | .939 | 75.10% |
| 5 | 400 | 1000 | 500 | .835 | 73.42% |
| 6 | 400 | 1000 | 1000 | .922 | 75.64% |
| 7 | 500 | 600 | 500 | .664 | 69.93% |
| 8 | 500 | 600 | 1000 | .567 | 68.08% |
| 9 | 500 | 800 | 500 | .734 | 73.39% |
| 10 | 500 | 800 | 1000 | .623 | 71.69% |
| 11 | 500 | 1000 | 500 | .739 | 73.39% |
| 12 | 500 | 1000 | 1000 | .489 | 66.04% |
| 13 | 600 | 600 | 500 | .687 | 62.32% |
| 14 | 600 | 600 | 1000 | .683 | 64.98% |
| 15 | 600 | 800 | 500 | .695 | 62.33% |
| 16 | 600 | 800 | 1000 | .703 | 58.40% |
| 17 | 600 | 1000 | 500 | .712 | 62.56% |
| 18 | 600 | 1000 | 1000 | .743 | 67.37% |

TABLE II

FAULT DETECTIONS AND FALSE ALARMS RATES ON TE PROCESS

| | Fault Detection Rates | | False Alarm Rates | | g-means | |
|---|---|---|---|---|---|---|
| Fault | GAN model | OCSVM | GAN model | OCSVM | GAN model | OCSVM |
| 1 | **99.625%** | 99.5% | 3.125% | **0.625%** | 98.24% | **99.44%** |
| 2 | 98.5% | 98.5% | 1.25% | **0%** | 98.62% | **99.25%** |
| 3 | **10.375%** | 7.625% | **7.5%** | 10.625% | **30.98%** | 26.11% |
| 4 | **56.25%** | 50.375% | 3.75% | **0.625%** | **73.58%** | 70.75% |
| 5 | **32.375%** | 30.5% | 3.75% | **0.625%** | **55.82%** | 55.05% |
| 6 | 100% | 100% | 0% | 0% | 100% | 100% |
| 7 | **100%** | 99.625% | 1.875% | **0%** | 99.06% | **99.81%** |
| 8 | **97.875%** | 97.375% | 1.25% | **0%** | 98.31% | **98.68%** |
| 9 | **8.625%** | 7.125% | **9.375%** | 21.875% | **27.96%** | 23.59% |
| 10 | 50.875% | **53.25%** | 0.625% | **0%** | 71.10% | **72.97%** |
| 11 | **58%** | 54.75% | 2.5% | **0.625%** | **75.20%** | 73.76% |
| 12 | **98.75%** | 98.625% | **4.375%** | 13.125% | **97.17%** | 92.56% |
| 13 | **95%** | 94.875% | 1.875% | 1.875% | **96.55%** | 96.49% |
| 14 | 100% | 100% | 1.875% | **0%** | 99.06% | **100%** |
| 15 | 12.5% | **14%** | 2.5% | **0%** | 34.91% | **37.42%** |
| 16 | 34.375% | **36.375%** | 23.75% | **20.625%** | 51.20% | **53.73%** |
| 17 | **91.125%** | 87.25% | 1.875% | **0%** | **94.56%** | 93.41% |
| 18 | **90.375%** | 90.125% | 1.875% | **0%** | 94.17% | **94.93%** |
| 19 | **11.875%** | 3.75% | 0.625% | 0.625% | **34.35%** | 19.30% |
| 20 | **58.375%** | 52.75% | 0% | 0% | **76.40%** | 72.63% |
| 21 | **49.875%** | 41.875% | **5.625%** | 6.25% | **68.61%** | 62.66% |
| Average | **64.51%** | 62.78% | 3.77% | **3.69%** | **75.04%** | 73.45% |

- Pri

- Conclusions & future work

- Conclusions & future work
- experience

- Conclusions & future work
- experience
- **Models** are anything an agent

- Conclusions & future work
- experience
- **Models** are anything an agent
- **Distributed models**