Improving Credit Card Fraud Detection using Generative Adversarial Networks (GAN)

6501 Capstone Group 4

Team Member:
Hao Ning, Jun Ying

Outline

- Problem Statement
- Exploratory Data Analysis (EDA)
- GAN Theory
- Base Model
- GAN & Performance Evaluation
- WGAN & Performance Evaluation
- WGAN-GP & Performance Evaluation
- GAN with Pre-Train Performance
- BEGAN & Performance Evaluation
- BAGAN & Performance Evaluation
- Conclusions

Problem Statement

- Credit Card Fraud Detection
  - Imbalanced dataset
  - European transactions within two days in September 2013
  - Kaggle Dataset: https://www.kaggle.com/mlg-ulb/creditcardfraud
- Oversampling & Undersampling
  - Biased towards the mainstream category
- Oversampling with GANs
  - How to evaluate the quality of generated data
- GANs Comparison
  - WGAN, WGAN_GP, BAGAN, BEGAN

EDA

- About Dataset: 284,807 transactions
  - ```
    Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
           'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
           'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
           'Class'],
       Time        V1        V2        V3  ...       V27       V28  Amount  Class
    0   0.0 -1.359807 -0.072781  2.536347  ...  0.133558 -0.021053  149.62      0
    1   0.0  1.191857  0.266151  0.166480  ... -0.008983  0.014724    2.69      0
    2   1.0 -1.358354 -1.340163  1.773209  ... -0.055353 -0.059752  378.66      0
    3   1.0 -0.966272 -0.185226  1.792993  ...  0.062723  0.061458  123.50      0
    4   2.0 -1.158233  0.877737  1.548718  ...  0.219422  0.215153   69.99      0
    ```

    There are 30 features and 1 class (normal:0, fraud:1) & no null values.

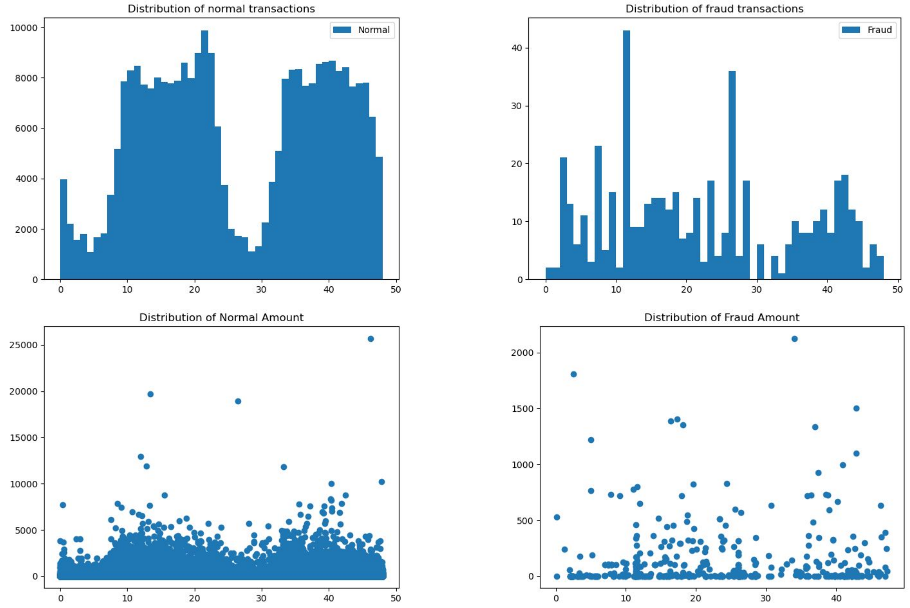
    Features V1 - V28 are the principal components obtained with PCA
- Imbalance Dataset：Only 0.173% data are fraud transactions.
  - ```
    The amounts of normal transactions (class 0) & fraud transactions (class 1)
    0    284315
    1       492
    ```


THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

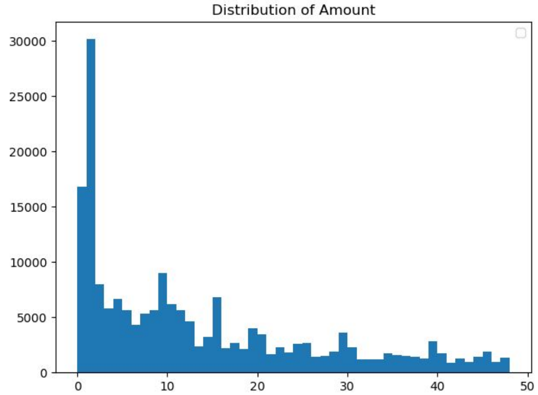
EDA

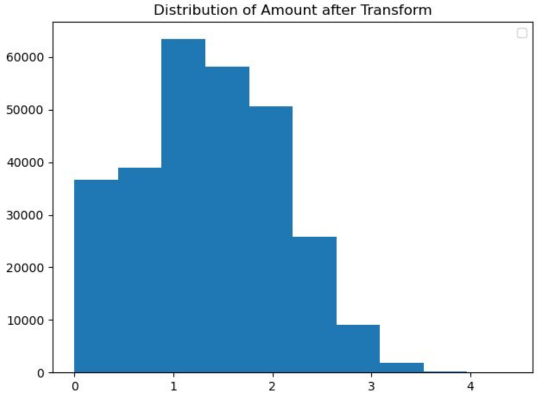
- Normal Transaction versus Fraud Transaction



EDA

- Data Preprocessing
  - ‘Time’ is sorted in time series in seconds change to hours
    - df['Time'] = (df['Time'].values / 3600)
  - ‘Amount’ is larger than other features
    - df['Amount'] = np.log10(df['Amount'].values + 1)



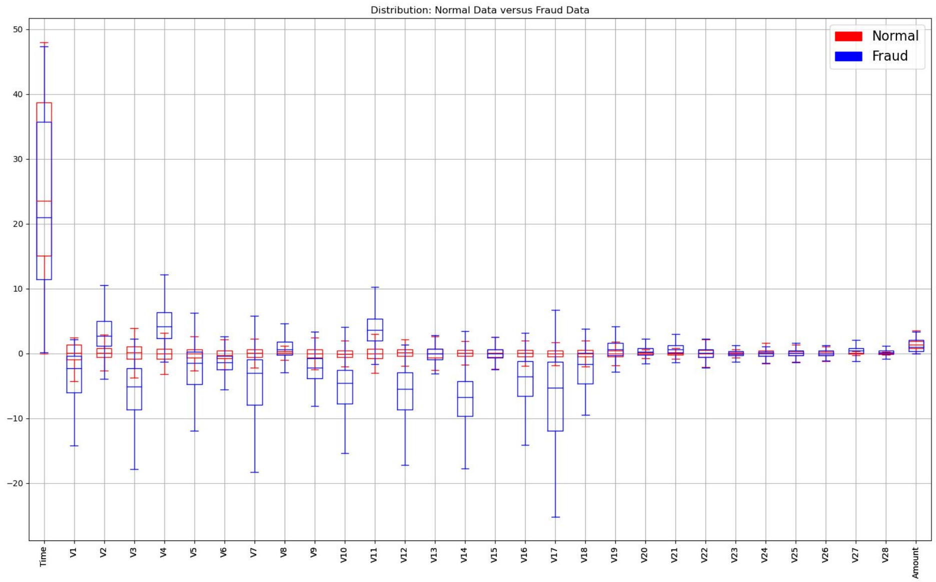


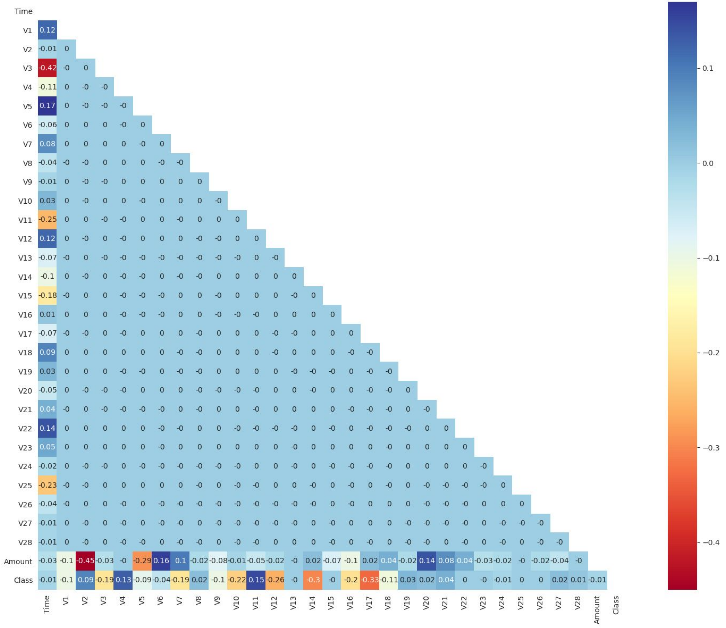
THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

EDA

- Data Distribution & Correlation of the Features



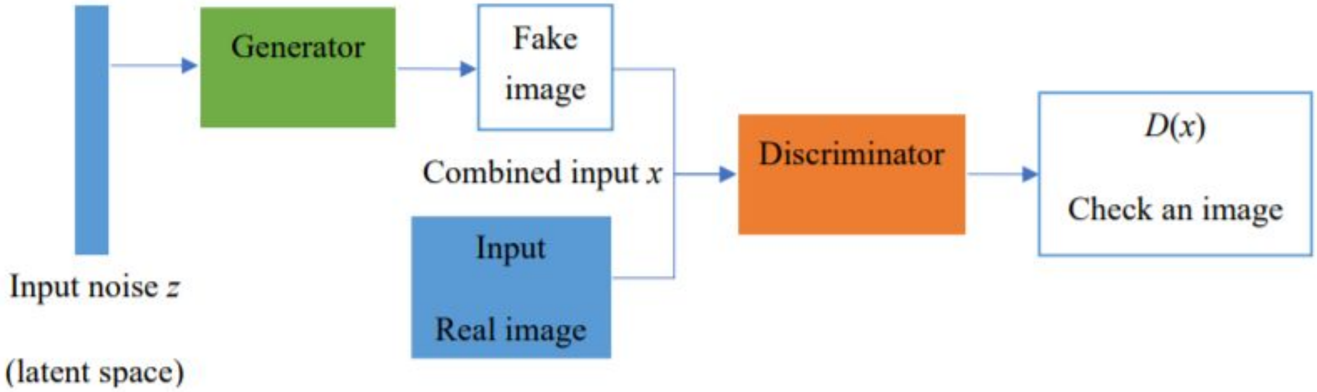


THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

GAN Theory

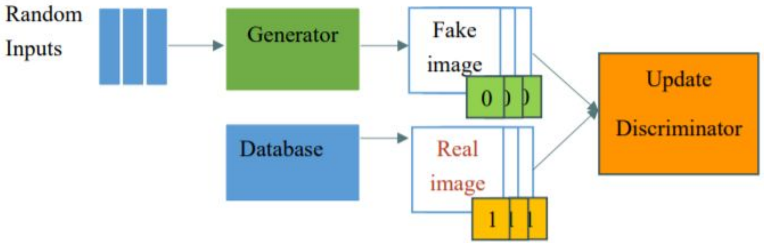
- Generative Adversarial Networks (GAN) is a cutting-edge technique of deep neural networks, which was first come up by Ian Goodfellow in 2014.

$$min_D max_G V(G, D) = \mathbb{E}_x \log(D(x)) + \mathbb{E}_z \log\left(D(G(z))\right)$$



GAN Theory

- GAN is a generative model in which two neural networks (generator and discriminator) compete and improve each other.
    - Training Discriminator



    - Training Generator

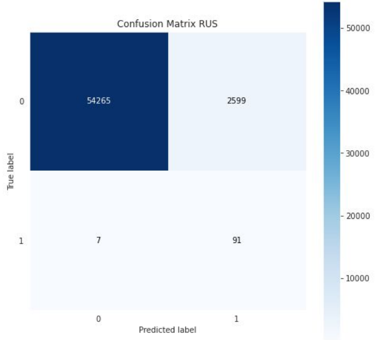
- Challenge
    - How to verify that the generated data meets the requirements?

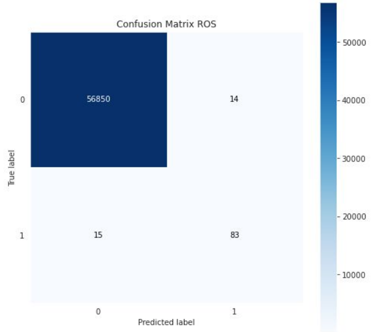
THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Base Model

- Train/Test Split & Stratified
  - Train 80% 227845 (394 fraud)
  - Test 20% 56962 (98 fraud)
- Random Under Sampling (RUS) & Random Over Sampling (ROS)
- GridsearchCV XGBoost
- Predict with best_params
- Test performance evaluation

	RUS 1 394 0 394	ROS/Base Model 1 227451 0 227451
Accuracy	0.954250	0.999491
Precision	0.033829	0.855670
Recall	0.928571	0.846939
F1 score	0.065280	0.851282
ROC AUC score	0.941433	0.923346





THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

GAN

- Differences from vanilla GAN
  - Original GAN deal with images, but we are dealing with tabular data, tahn is removed
  - Batch normalization removed
- Noise = 32, len(features) = 30

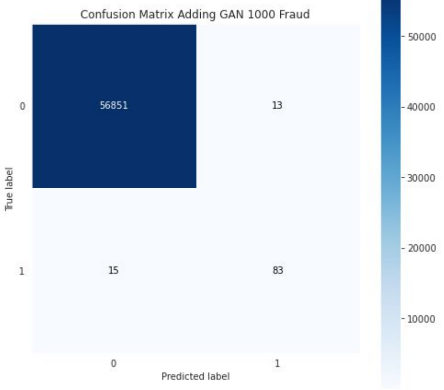
Generator Input	Generator output
(32, 1) **Noise**	(64,1)
(64, 1)	(128, 1)
(128, 1)	(256, 1)
(256, 1)	(30, 1) **Features**

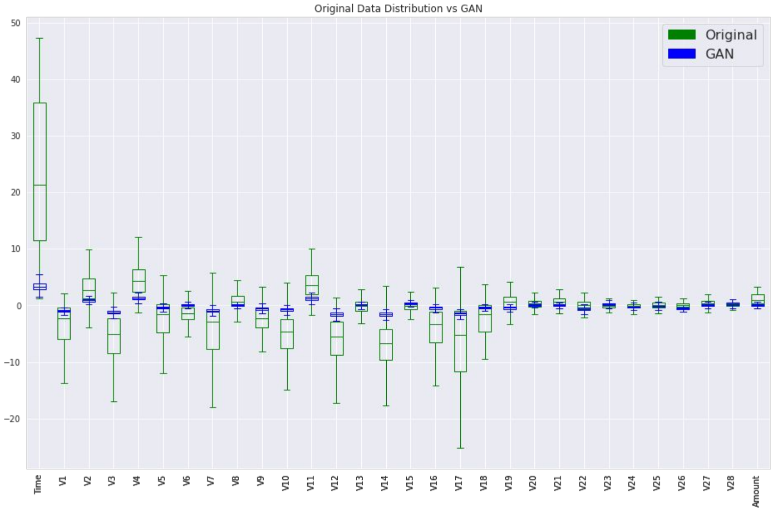
Discriminator Input	Discriminator Output
(30, 1) **Features**	(256,1)
(256, 1)	(128, 1)
(128, 1)	(64, 1)
(64, 1)	(1, 1) **label**

GAN Performance

- Slight improvement, adding 1000 generated fraud
- Low spectrum of data
  - Mode collapse

	Base	GAN
Accuracy	**0.999491**	**0.999491**
Precision	**0.855670**	**0.864583**
Recall	**0.846939**	**0.846939**
F1 score	**0.851282**	**0.855670**
ROC AUC score	**0.923346**	**0.923355**





THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

WGAN

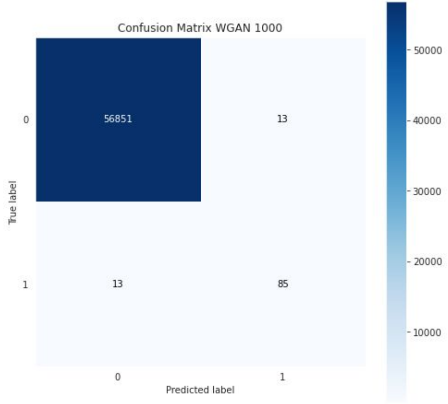
- Why WGAN
  - Prevent gradient vanishing & mode collapse in vanilla GAN
  - Evaluate the difference between real and generated samples with wasserstein distance, using a score rather than label
- Implementation compared to GAN
  - Loss function, no log
  - D: no sigmoid, and train D more than G
  - Clip the weight of D at (-0.01,0.01)
  - Use RMSProp

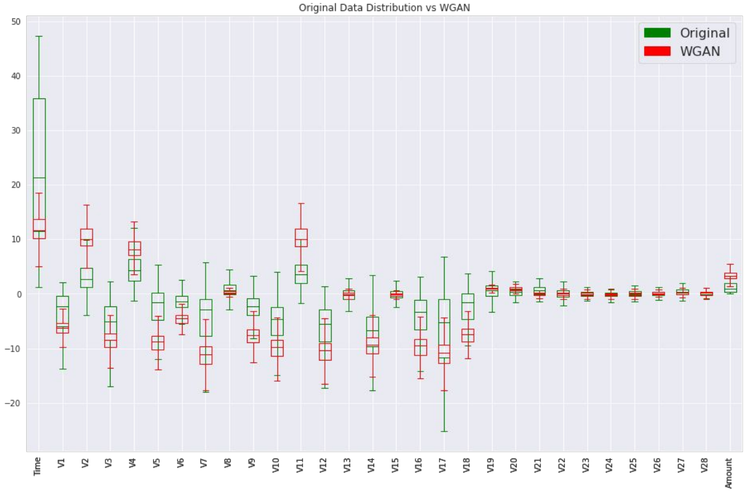
WGAN	D loss: $\tilde{V} = \frac{1}{m}\sum_{i=1}^{m}\left(D(x^i) - D\left(G(z^i)\right)\right)$	G loss: $\tilde{V} = \frac{1}{m}\sum_{i=1}^{m} D\left(G(z^i)\right)$

WGAN Performance

- Obvious improvement, adding 1000 generated fraud
- Wider spectrum of data
  - Overlap well with the original data
  - No mode collapse like GAN

	GAN	WGAN
Accuracy	0.999508	0.999544
Precision	0.864583	0.867347
Recall	0.846939	0.867347
F1 score	0.855670	0.867347
ROC AUC score	0.923355	0.933559

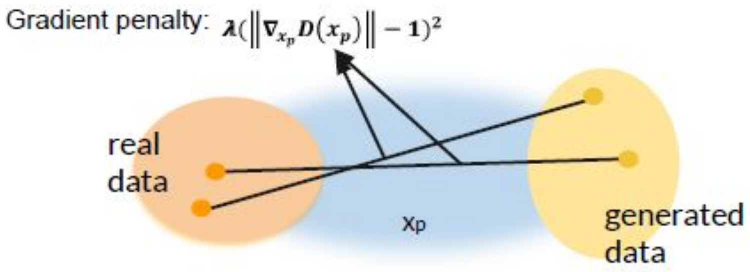




THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

WGAN_GP

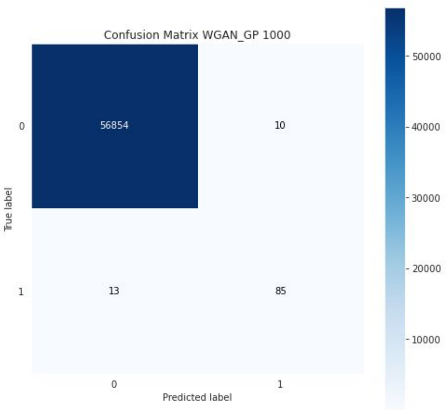
- Why WGAN_GP $|f(x_1) - f(x_2)| \leq K|x_1 - x_2|.$
    - A differentiable function is 1-Lipschitz function if and only if it has gradients with norm at most 1 everywhere
    - Hard to get gradients with norm at most 1 everywhere, interpolate between real and generated samples instead
    - Gradient penalty instead of weight clipping in WGAN

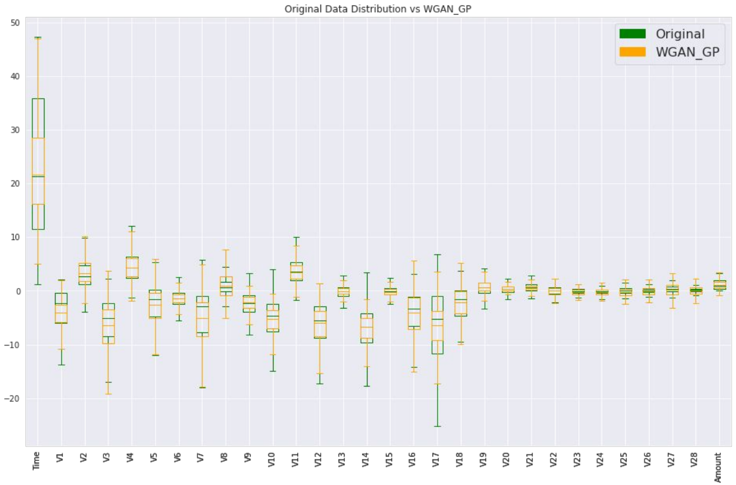


WGAN_GP Performance

- ADAM(lr=0.00001), batch_size=128, gp_lambda=5
- Obvious improvement, adding 1000 generated fraud
- Wider range & excellent overlap

	WGAN	**WGAN_GP**
Accuracy	0.999544	**0.999596**
Precision	0.867347	**0.894737**
Recall	0.867347	**0.867347**
F1 score	0.867347	**0.880829**
ROC AUC score	0.933559	**0.933586**



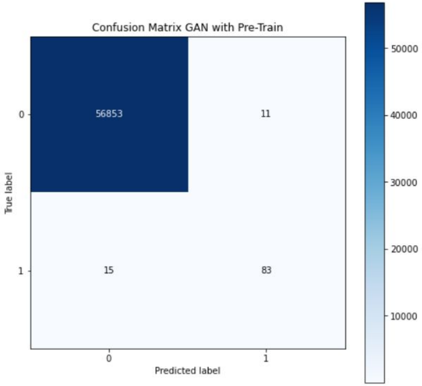


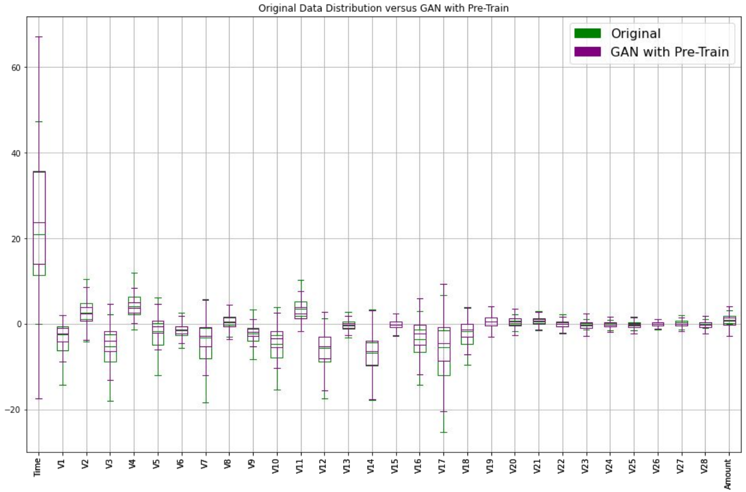
THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

GAN with Pre-Train Performance

- Using the parameter of trained autoencoder
- Better than Original GAN, slightly worse than WGAN
- Wider spectrum of data
  - Overlap well with the original data

	GAN	GAN + AE
Accuracy	0.999508	0.999544
Precision	0.864583	0.882979
Recall	0.846939	0.846939
F1 score	0.855670	0.864583
ROC AUC score	0.923355	0.923373

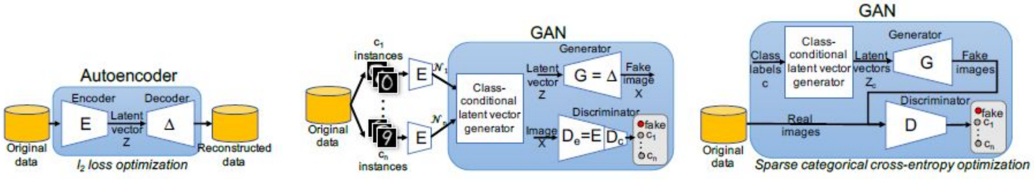




THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

BAGAN

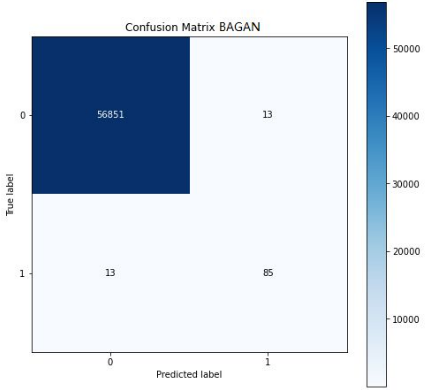
- Why BAGAN:
  - An augmentation tool
  - Learn from all data
- Compared to original GAN:
  - Discriminator output class label c or the label fake.
  - Coupling GAN and autoencoding techniques.

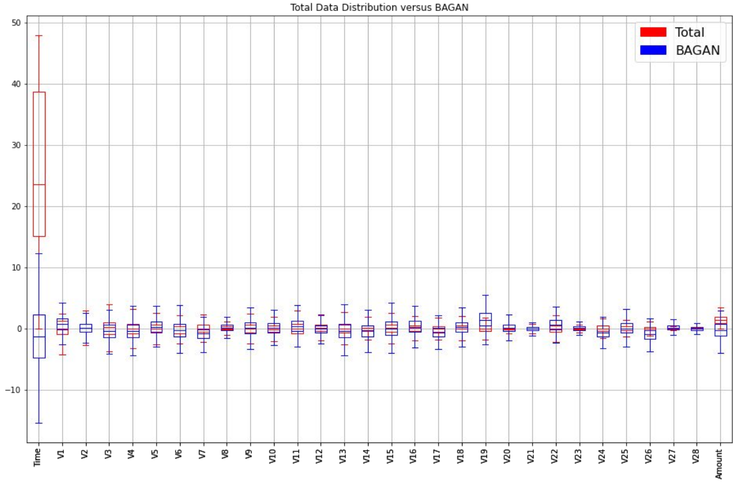

(a) Autoencoder training. (b) GAN initialization. (c) GAN training.

BAGAN Performance

- Performance better than GAN
- Generated fraud data same as the total dataset.
  - Learn too much from normal data
  - No similarity between these two classes

	GAN	BAGAN
Accuracy	0.999508	0.999544
Precision	0.864583	0.867347
Recall	0.846939	0.867347
F1 score	0.855670	0.867347
ROC AUC score	0.923355	0.933559





THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

BEGAN

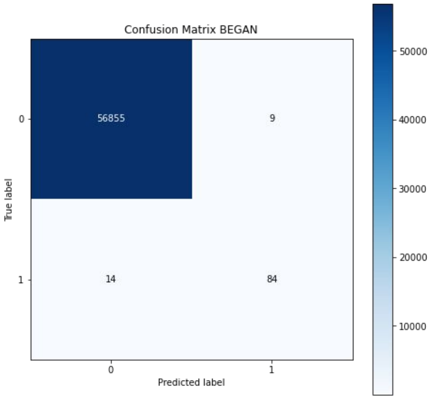
- Why BEGAN:
  - A GAN with simple yet robust architecture, standard training procedure with fast and stable convergence.
  - An equilibrium concept that balances the power of the discriminator against the generator.
  - Use Proportional Control Theory to maintain the equilibrium
  - Using a variable kt ∈ [0, 1] to control how much emphasis is put on L(G(zD)) during gradient descent.

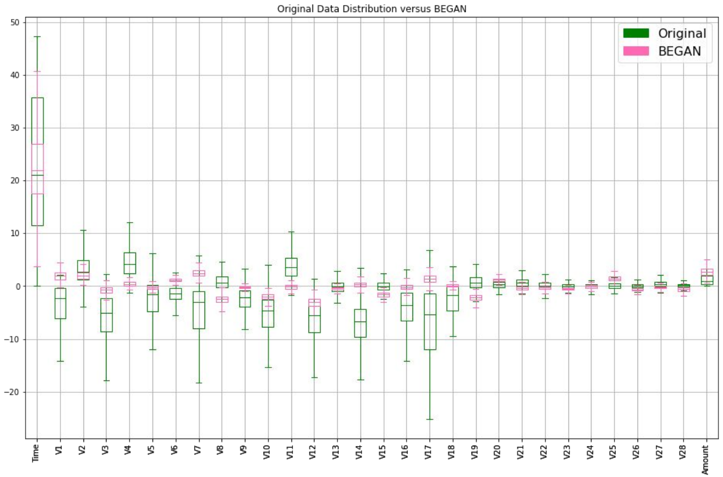
$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t.\mathcal{L}(G(z_D)) & \text{for } \theta_D \\ \mathcal{L}_G = \mathcal{L}(G(z_G)) & \text{for } \theta_G \\ k_{t+1} = k_t + \lambda_k(\gamma\mathcal{L}(x) - \mathcal{L}(G(z_G))) & \text{for each training step } t \end{cases}$$

BEGAN Performance

- Performance better than BAGAN & WGAN
- Cover most of fraud data spectrum

	BAGAN	BEGAN
Accuracy	0.999544	0.999596
Precision	0.867347	0.903226
Recall	0.867347	0.857143
F1 score	0.867347	0.879581
ROC AUC score	0.933559	0.928492





Comparison of all Models

	Base	GAN	WGAN	WGAN_GP Best 1	GAN + AE	BAGAN	BEGAN Best 2
Accuracy	0.999491	0.999491	0.999544	**0.999596**	0.999544	0.999544	**0.999596**
Precision	0.855670	0.864583	0.867347	**0.894737**	0.882979	0.867347	**0.903226**
Recall	0.846939	0.846939	0.867347	**0.867347**	0.846939	0.867347	**0.857143**
F1 score	0.851282	0.855670	0.867347	**0.880829**	0.864583	0.867347	**0.879581**
ROC AUC score	0.923346	0.923355	0.933559	**0.933586**	0.923373	0.933559	**0.928492**

Conclusions

- GANs work well with tabular data with proper modification
  - Use data distribution to check the quality of generated data
- Vanilla GAN slightly improved fraud detection
  - Low spectrum data
- Improved GAN models showed better fraud detection performance
  - Wider range & better overlap
- **WGAN_GP** and **BEGAN** perform best among all GANs
- Using GAN as an oversampling strategy has great potential in credit card fraud detection and extremely imbalanced dataset

References

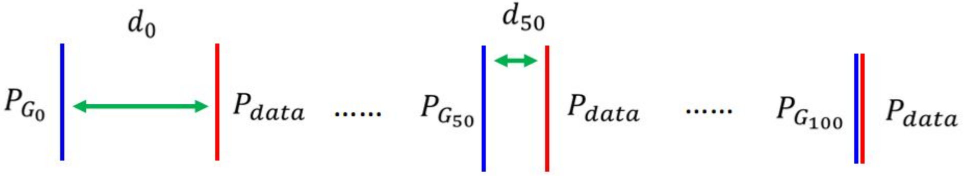
1. GAN Generative Adversarial Networks
2. Wasserstein GAN [1701.07875] Wasserstein GAN
3. Improved Training of Wasserstein GANs https://arxiv.org/pdf/1704.00028
4. BAGAN [1803.09655] BAGAN: Data Augmentation with Balancing GAN
5. BEGAN: Boundary Equilibrium Generative Adversarial Networks [1703.10717] BEGAN: Boundary Equilibrium Generative Adversarial Networks
6. https://github.com/eriklindernoren/Keras-GAN
7. https://github.com/IBM/BAGAN

Thank you!

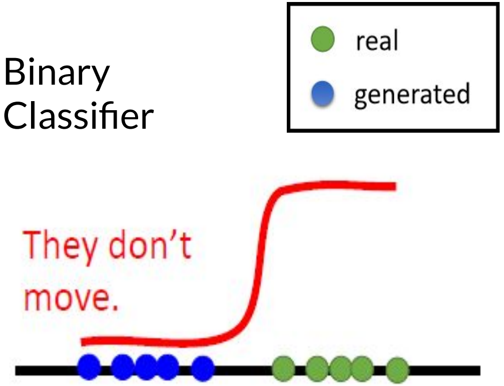
Q&A

Supplemental Materials

- JS Convergence vs Wasserstein
  - Loss function of GAN is equivalent to JS convergence(D at optimality)
  - JS is always log2 if two distributions do not overlap (0 gradient)



Binary Classifier



$JS(P_{G_0}, P_{data}) = log2$ $\qquad$ $JS(P_{G_{50}}, P_{data}) = log2$ $\qquad$ $JS(P_{G_{100}}, P_{data}) = 0$

$W(P_{G_0}, P_{data}) = d_0$ $\qquad$ $W(P_{G_{50}}, P_{data}) = d_{50}$ $\qquad$ $W(P_{G_{100}}, P_{data}) = 0$