MAT407 Machine Learning
Projet Final : Détection d'Anomalie avec des Données Synthétiques Générées par GAN

Ömer Tüfekçi - Umut B. Yıldırım

24.05.2024

Contenu

- Objective
- Méthodologie
- Définition & Architecture de GAN
- Codage en parties
- Défis principaux
- Résultats et comparaison
- Conclusion

Objective

- Appliquer le GAN pour l'augmentation des données dans un contexte d'apprentissage automatique.
- Améliorer la performance de la détection de fraude grâce à l'intégration de données synthétiques générées par GAN.
- Valider l'efficacité des GAN pour améliorer les modèles d'apprentissage automatique.
- Comparer la performance des classificateurs entraînés avec et sans les données augmentées par GAN.

Méthodologie / Bibliothéques

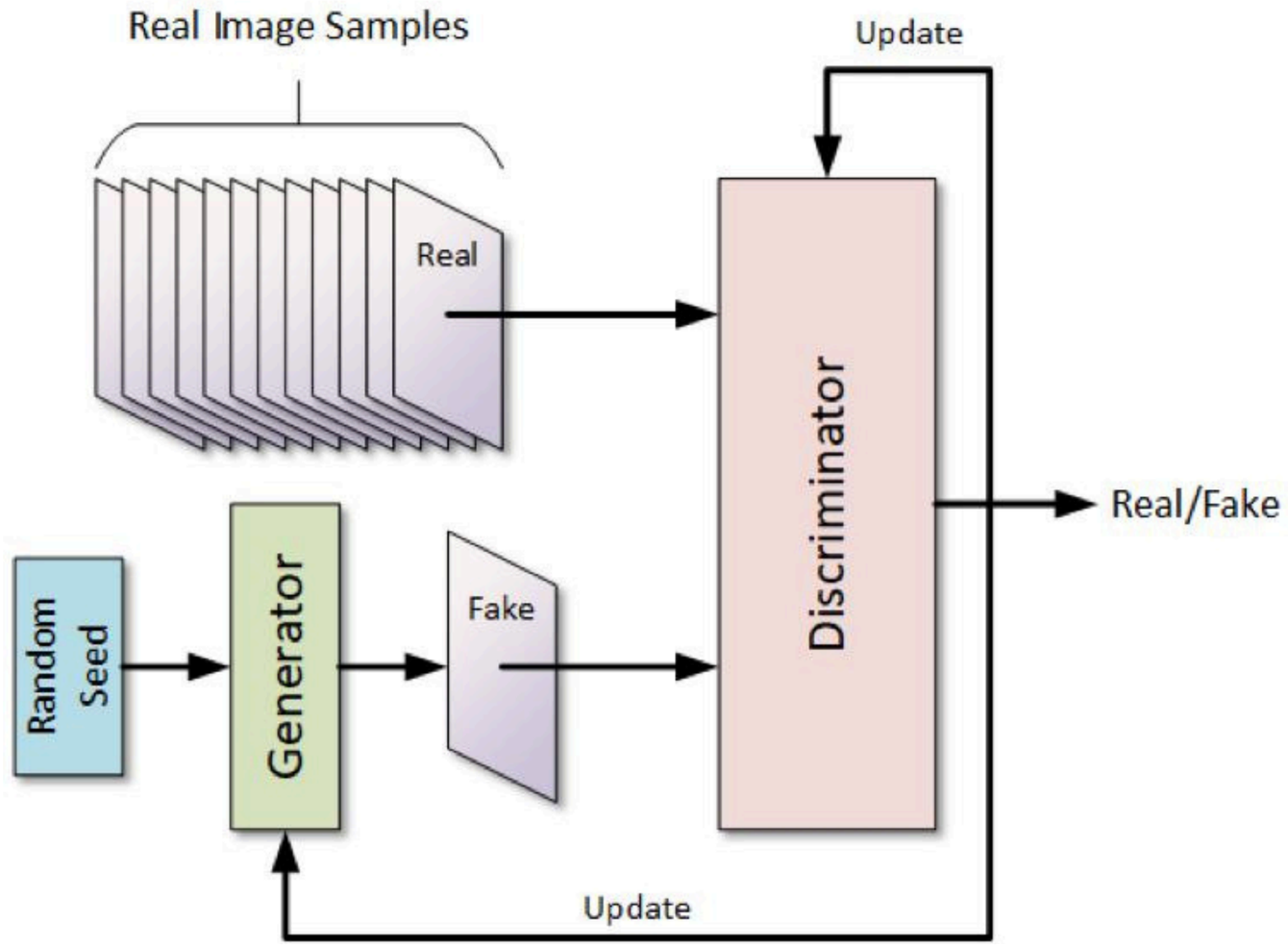
- Développement et Formation d'un GAN
- Entraînement du GAN
- Détection d'Anomalies
- Évaluation de la Performance
- Évaluation de la Performance
- Amélioration et Optimisation

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, BatchNormalization, LeakyReLU
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import BinaryCrossentropy
from tensorflow.keras.metrics import Precision, Recall
import tensorflow.keras.backend as K
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

Qu'est-ce qu'un GAN ?

- GAN est composé de deux réseaux neuronaux : le générateur et le discriminateur.
- Générateur :
  - Produit des données synthétiques à partir d'un bruit aléatoire.
  - Tente de créer des données réalistes pour tromper le discriminateur.
- Discriminateur :
  - Évalue les données et distingue les données réelles des données synthétiques.
  - Retourne un score de probabilité indiquant l'authenticité des données.
- Entraînement :
  - Processus antagoniste où le générateur et le discriminateur s'améliorent mutuellement.
  - Le générateur tente de tromper le discriminateur, tandis que le discriminateur devient meilleur pour détecter les faux.

Architecture de GAN



Codage I : Générateur & Discriminateur

```python
def build_generator(latent_dim, output_dim):
    model = Sequential()
    model.add(Dense(256, input_dim=latent_dim))
    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(output_dim, activation='tanh'))
    return model
```

- Couches Denses
- Activation LeakyReLU
- Batch Normalization
- Activation Tanh
- Dropout
- Activation Sigmoid
- Fonction de Perte
- Optimiseur Adam

```python
def build_discriminator(input_dim):
    model = Sequential()
    model.add(Dense(512, input_dim=input_dim))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.4))
    model.add(Dense(256))
    model.add(LeakyReLU(alpha=0.2))
    model.add(Dropout(0.4))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer=Adam(learning_rate=0.0002, beta_1=0.5), loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

Codage II : Construction et Entraînement du GAN

- Gel du Discriminateur
- Combinaison des Modèles
- Entraînement

```python
def train_gan(gan, generator, discriminator, X_train, epochs, batch_size, latent_dim):
    for epoch in range(epochs):
        # Select a random batch of real data
        idx = np.random.randint(0, X_train.shape[0], batch_size)
        real_data = X_train[idx]

        # Generate a batch of fake data
        noise = np.random.normal(0, 1, (batch_size, latent_dim))
        fake_data = generator.predict(noise)

        # Create labels for real and fake data
        real_labels = np.ones((batch_size, 1)) * 0.9  # Label smoothing
        fake_labels = np.zeros((batch_size, 1))

        # Train the discriminator
        d_loss_real = discriminator.train_on_batch(real_data, real_labels)
        d_loss_fake = discriminator.train_on_batch(fake_data, fake_labels)
        d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)

        # Train the generator
        noise = np.random.normal(0, 1, (batch_size, latent_dim))
        valid_y = np.ones((batch_size, 1))
        g_loss = gan.train_on_batch(noise, valid_y)

        # Print the progress
        if epoch % 100 == 0:
            print(f"{epoch}/{epochs} [D loss: {d_loss[0]} | D accuracy: {100 * d_loss[1]}] [G loss: {g_loss}]")
```

```python
latent_dim = 100
generator = build_generator(latent_dim, X_train.shape[1])
discriminator = build_discriminator(X_train.shape[1])
gan = build_gan(generator, discriminator)
train_gan(gan, generator, discriminator, X_train.values, epochs=1000, batch_size=64, latent_dim=latent_dim)
```

Codage III: Données Synthétiques

- Génération Synthétique
- Combinaison des Données
- Pondération des Classes
- Validation

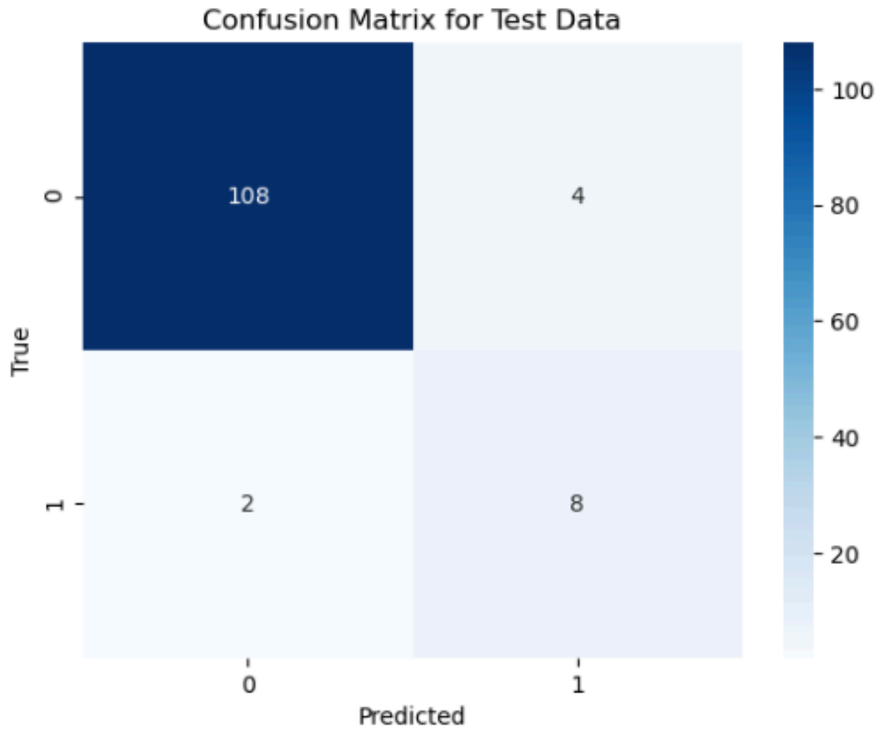
```python
def generate_synthetic_data(generator, num_samples, latent_dim):
    noise = np.random.normal(0, 1, size=(num_samples, latent_dim))
    synthetic_data = generator.predict(noise)
    return synthetic_data
```

```python
def prepare_test_data(X_test, synthetic_data, y_test):
    synthetic_labels = np.ones((synthetic_data.shape[0],))
    X_combined = np.vstack((X_test, synthetic_data))
    y_combined = np.concatenate((y_test, synthetic_labels))
    return X_combined, y_combined
```

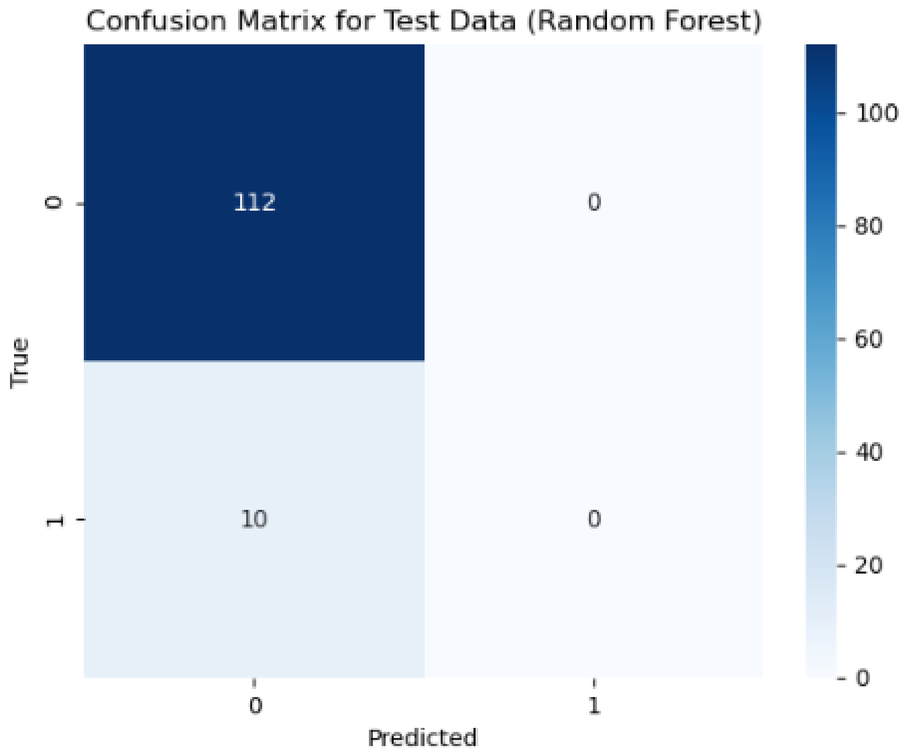
```python
X_combined_train, X_combined_val, y_combined_train, y_combined_val = train_test_split(
    X_combined, y_combined, test_size=0.2, random_state=42, stratify=y_combined)
```

```python
class_weight = {0: 1., 1: 10.}
```

Résultats et comparaison



	precision	recall	f1-score	support
0	0.98	0.96	0.97	112
1	0.67	0.80	0.73	10
accuracy			0.95	122
macro avg	0.82	0.88	0.85	122
weighted avg	0.96	0.95	0.95	122



	precision	recall	f1-score	support
0	0.92	1.00	0.96	112
1	0.00	0.00	0.00	10
accuracy			0.92	122
macro avg	0.46	0.50	0.48	122
weighted avg	0.84	0.92	0.88	122

Certaines défis principaux

- Déséquilibre des Données
- Choix de Classificateur
- Division des données
- Overfitting (Surapprentissage)
- Stabilité de l'Entraînement du GAN
- Performance de Détection des Anomalies

Conclusion

- Le modèle GAN a démontré une capacité supérieure à détecter les anomalies par rapport aux méthodes traditionnelles comme la forêt aléatoire.
- Ce projet a démontré l'efficacité des GANs pour la détection des anomalies, offrant une solution robuste pour des applications nécessitant une détection précise.
- Les résultats obtenus fournissent une base solide pour des travaux futurs dans l'application des GANs à d'autres domaines de la détection des anomalies.
- Dans les travaux futurs on doit, encore, améliorer la capacité du discriminateur à classifier avec précision la classe minoritaire.

Bibliographie

- Saracco, Francesco. "Detecting the Unseen: Anomaly Detection with GANs." Medium, Data Reply IT | DataTech, 12 décembre 2023. Disponible à : *https://medium.com/data-reply-it-datatech/detecting-the-unseen-anomaly-detection-with-gans*

- Prince, Simon J.D. Understanding Deep Learning. Deep Learning Publications, 2020, pp. 275-302.