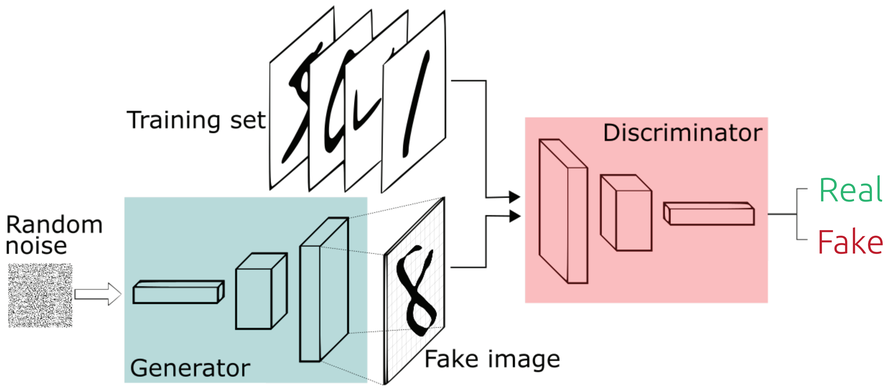
TAIGAN

Towards Accessible Improved Generative Adversarial Networks

Les GAN

GAN : type de réseaux de neurones permettant de générer des données, notamment des images. Il s'agit du domaine de recherche en IA le plus actif en ce moment

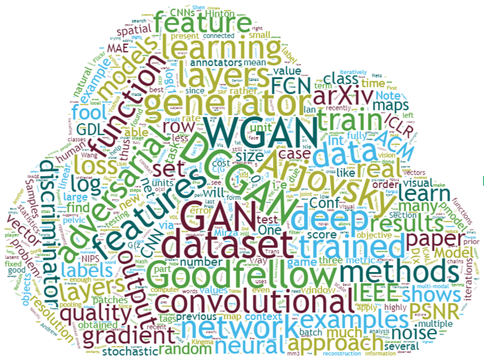
Dans un GAN, le générateur génère des images et le discriminateur devine si les images qu'on lui montre sont vraies ou fausses.
Avec assez de données, le générateur va apprendre à générer des images dans le style de ce qu'il a déjà vu.



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

Les GAN sont alors parfaits pour réaliser de l'augmentation de données ! Notamment dans le domaine médical pour améliorer les systèmes à base d'IA. Malgré cela, ils restent encore très durs à utiliser ...

Nos Résultats



Implémentation et analyse de l'état de l'art des GAN dans un guide pour faciliter leur usage

Application de ces méthodes au domaine de l'imagerie médicale

Et plus encore....

Une littérature dense et variée ... Un travail de réflexion ... Une démarche d'innovation ...

Guide sur les GAN

In order to calculate the penalty, we first calculate a random weighted average between real and fake samples $\hat{x}$ and then we calculate the following penalty

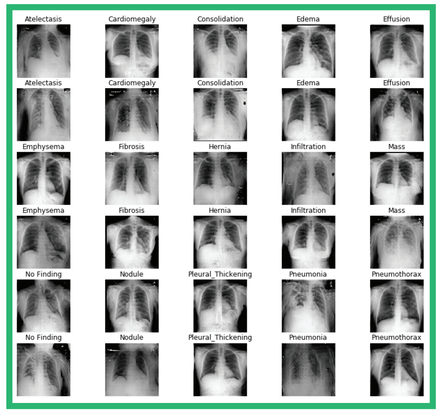
$$GP = \lambda(\|\nabla_{\hat{x}} D_w(\hat{x})\|_2 - 1)^2$$

Then to adapt the training loop from WGAN to WGAN-GP we only have to add this penalty to our critic's loss (discriminator in classical GANs).

```
# Notice the use of `tf.function`
# This annotation causes the function to be "compiled".
@tf.function
def train_step_d(images, noise, labels):
  with tf.GradientTape() as disc_tape:
        generated_images = generatGc([noise,labels] , training=True)

        #==============================================
        #=                                            =
        #=                Gradient Penalty            =
        #=                                            =
        #==============================================

        with tf.GradientTape() as gp_tape:
          epsilon = tf.random.uniform([BATCH_SIZE, 1, 1, 1])
          x_hat = epsilon*images + (1-epsilon)*generated_images
          gp_tape.watch(x_hat)
          D_hat = discriminator([x_hat,labels], training=True)

        gradients = gp_tape.gradient(D_hat, x_hat)[0]
        gradients_norm = tf.sqrt(tf.reduce_sum(tf.square(gradients), axis=np.arange(1,len(gra
        penalty = tf.reduce_mean(LAMBDA * tf.square(gradients_norm - 1.0))

        #==============================================
        #=                                            =
        #=              Wasserstein Loss              =
        #=                                            =
        #==============================================


        real_output = discriminator([images, labels], training=True)
        fake_output = discriminator([generated_images, labels], training=True)

        disc_loss = tf.reduce_mean(-real_output + fake_output)

        #==============================================
        #=                                            =
        #=             Total Critic Loss              =
        #=                                            =
        #==============================================
        loss = disc_loss + penalty

        gradients_of_discriminator = disc_tape.gradient(loss, discriminator.trainable_variabl
        discriminator2_optimizer.apply_gradients(zip(gradients_of_discriminator, discriminator

        return disc_loss
```

Guide sur les dernières avancées du domaine pour faciliter l'utilisation des GAN

Application



Génération d'images de scanner pour 15 pathologies afin d'améliorer leur détection automatique

Quentin GARRIDO, Anthony LAFOURCADE, Raphaël LAPERTOT, Josselin LEFEVRE - E3 INF
Suiveur : Laurent NAJMAN

une école de la CCI PARIS ILE-DE-FRANCE