# Implémentation du réseau de neurones MONet

**Source :** Christopher P. Burgess, Loic Matthey, Nicholas Watters,Rishabh Kabra, Irina Higgins, Matt Botvinick, Alexander Lerchner, DeepMind, MONet: Unsupervised Scene Decomposition andRepresentation, URL : <https://arxiv.org/pdf/1901.11390v1.pdf>

## Architecture du code :

* main.py : appel les autres fichiers, affiche les résultats de l’entrainement et d’un appel simple du model
  + monet.py : crée le model, permet son entrainement
    - unet.py et vae.py : créent les 2 principaux composant du modèle monet. Ils sont appelés par leurs fonctions call() qui calcul l’output associée au tenseur reçu en entrée.
  + dataset\_generator.py : génère le dataset (tf.Data.Dataset) à partir des images stockées dans le dossier Data
* image\_generator.py : génère les images de formes simples en 2D

## Unet.py :

Hérite de la classe layer pour que ses poids puissent être transmis au model Monet et entrainés. Pour cette même raison il est obligatoire que chaque couche de Unet soit un attribut de la classe. De même pour le vae.

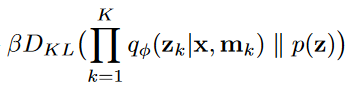
## Monet.py :

#### 1er terme de la loss (l1) :

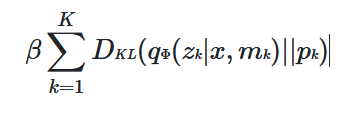
La méthode .prob(x) est utilisée par les tutoriels tensorflow dans le calcul de la vraisemblance en sortie du vae. Elle renvoie la densité de probabilité d’une distribution. Dans le cas de MONet elle est calculée pour chaque scope et pondérée par le masque issue de Unet. Ainsi la vraisemblance d’un pixel ne compte que pour le masque auquel il appartient.

#### Deuxième terme de la loss (l2) :

Le terme



Est transformé en



Il est calculé et ajouté automatiquement à la loss à chaque scope grâce au paramètre

activity\_regularizer=tfp.layers.KLDivergenceRegularizer(self.prior(), weight=1.0)

du dernier layer de l’inference\_net du VAE.

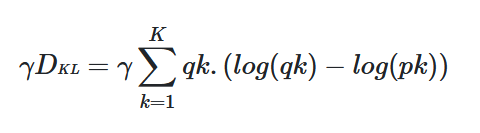
#### Troisième partie de la loss (l3) :



KL divergence entre :

* la distribution de probabilité Q d’appartenance d’un pixel à la couche k du masque créé par l’AN
* la distribution de probabilité P d’appartenance d’un pixel à la couche k du masque créé par le VAE

En développant la divergence on obtient :



On calcul la kl divergence entre les distributions de chaque pixel, donc 128\*128 Dkl.

Pour obtenir la loss je fais la moyenne de ces kl.

## Initialisation du réseau :

Tous les poids du réseau sont initialisés par une loi normale tronquée et un biais nul, conformément à l’article d’origine.

Pour Unet, l’écart type de cette loi normale est où Nin est le nombre de neurones en entrée de celui-ci.Cette initialisation est celle décrite dans l’article de Unet.

Pour le VAE, on utilise l’initialisation par défaut, qui est l’initialisation de Xavier uniforme.

Celle-ci consiste à initialiser les poids selon une loi normale centrée dont la variance est définie par 1/Nin .