DELIRES TP3: Synthèse de texture

Souhaib Attaiki

Mars 2019

1 Génération de texture de type Gatys

Question 1 Les couches qui ont été choisies pour caractériser la texture sont les couches de 1 à 5: 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1', 'conv5_1'.

Question 2 On remarque qu'en ajoutant des couches, la texture générée est de plus en plus satisfaisante et ressemble à la texture utilisée pour l'entraînement(voir figure 1).

On remarque aussi que la contribution des couches profondes à la génération de texture est plus importante que les premiers couche du réseau(voir figure 2). Les premières couches captent les structures locales de petite échelle, alors que les couches profondes captent les contours et structure de la texture à grande échelle.

Question 3 L'algorithme d'optimisation utilisée est AdamOptimizer.

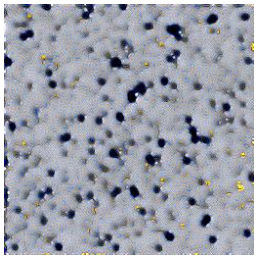
2 Génération de texture par Texturenet

Question 1 Un exemple d'overfeat possible est le *mode collapse*, c'est-à-dire que le réseau apprend à ne générer qu'une seule image (en multipliant le bruit d'entré par 0).

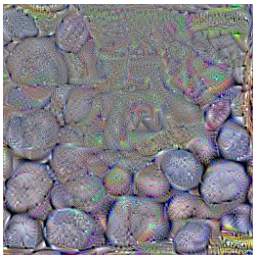
Question 2 Le réseau génératif est composé de plusieurs blocs convolutifs qui ont des tailles différentes (chaque bloc est composé de plusieurs couches convolutives, activation non linéaire *leaky_relu* et de BatchNorm), ces derniers sont alors concaténés en utilisant le module Join (voir

(a) 1 couche (b) 2 couches (c) 3 couches (d) 4 couches (e) 5 couches

Figure 1: Évolution de la texture générée en fonction du nombre de couches



(a) Couche conv1_1



(b) Couche conv5_1

Figure 2: Comparaison des contributions des différentes couches

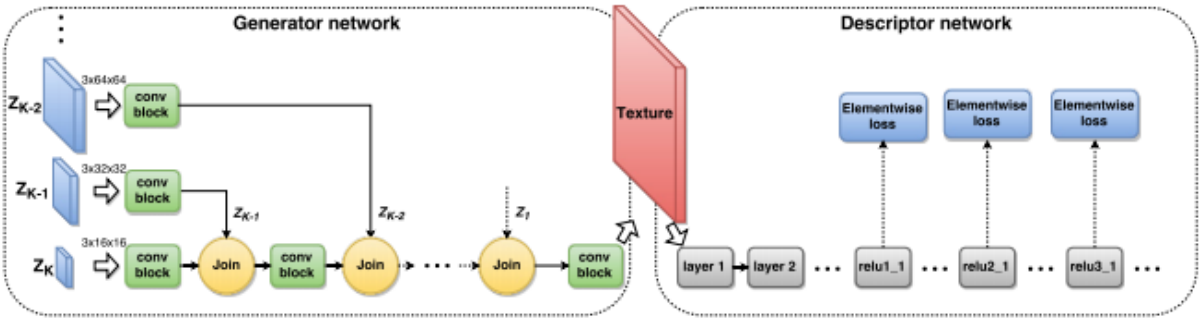


Figure 3: Architecture de TextureNet

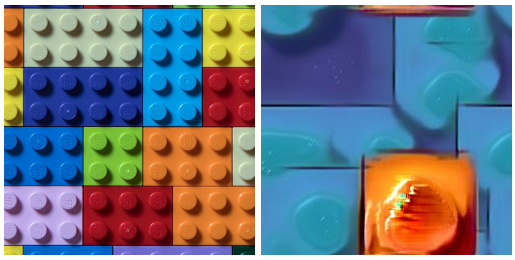
figure 3) qui fait un Upsampling et un BatchNorm. Finalement, une dernière convolution est appliqué avant de calculer la loss.

En utilisant la fonction *'parameter'* de PyTorch, on trouve que le nombre de paramètres est: 116835.

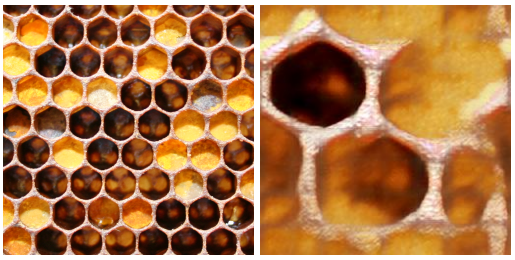
Question 3 On remarque que les résultats obtenus sont meilleurs que ceux de Gatys, cependant, les deux architectures trouve des difficultés à générer les textures complexes (voir figure 4). On remarque aussi que la texture générée représente le motif qui se répète dans chaque texture.

Question 4 On a utilisé une interpolation linéaire de la forme:

$$k[t,:,:,:] = (1-\lambda) * k[0,:,:,:] + \lambda * k[-1,:,:,:]$$



(a) Lego



(b) Pierza

Figure 4: Gauche: image d'entrée. Droite: image générée

avec $\lambda = \frac{t}{n_{sample}-1}$

Question 5 Le phénomène observé est dû à l'interpolation linéaire qu'on a implémenté pour générer les images du milieu. En effet, la somme des deux uniformes n'est pas une densité uniforme mais en forme de trapèze, et qui devient triangle en milieu de vidéo. Donc le résultat obtenu est conséquence du fait que le réseau est entraîné sur une densité uniforme, alors que la densité du milieu du film est une densité en forme de triangle.

Question 6 La fonction *redresse* permet de redresser la densité la distribution du milieu en la rapprochant de la distribution uniforme. Ceci améliore les textures générées au milieu de la vidéo.