



Descriptif:

**Classe Random pour initialiser les poids du filtre**

Encadrant: **Hugo Bolloré**

Par: **Aicha Maaoui**

Date: 30/12/2021

**Problématique**

Le but de cette partie est la description de la classe *Random\_weights* pour initialiser les poids des filtres appliqués à une image donnée à l'entrée.

**Classe Random weights**Generalités:

La première étape de l'utilisation de la méthode de **Gradient Descent** dans est la génération de nombres aléatoires pour les différentes valeurs de filtres utilisés.

On se propose d'utiliser dans cette partie des classes issues de la **bibliothèque standard (STL)** de *C++* (Standard Template Library, dont le préfixe est **std::**) [1].

La bibliothèque **Random** de STL permet de générer des variables aléatoires de loi donnée. On se propose d'utiliser la loi de densité de STL, **std::normal.distribution**.

Description de **std::normal.distribution** (*C++11*):

Il s'agit de la génération de nombres aléatoires selon la distribution de nombres aléatoires normale (ou gaussienne) [2], comme défini dans l'équation (1).

$$f(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-1}{2} \frac{x - \mu}{\sigma}^2 \quad (1)$$

avec  $\mu$  est la moyenne de distribution (mean) et  $\sigma$  est l'écart-type (standard deviation (stddev)).

L'utilisation de la distribution normale s'explique par le fait que c'est la génération d'un grand nombre de variables aléatoires indépendantes, suivant une même distribution.

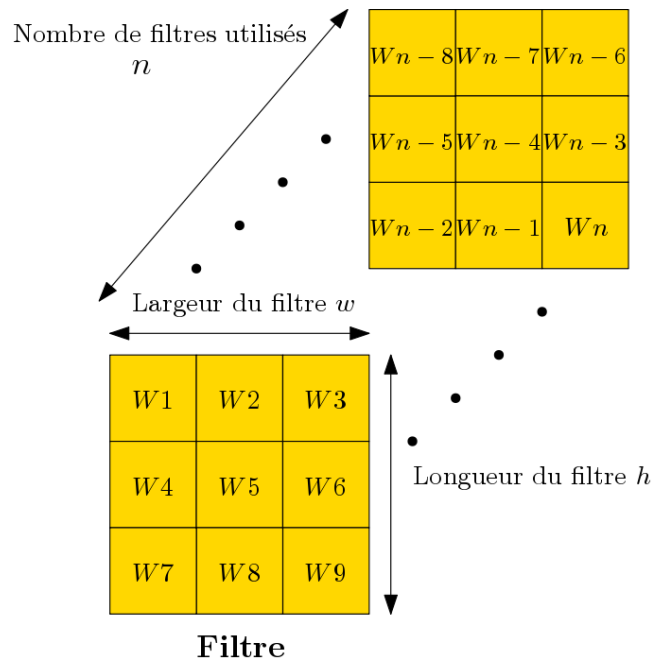


Figure 1: Filtres de CNN initialisés par des nombre aléatoires.

D'après la figure (1), il est clair que la classe `Random_weights` sera responsable de générer  $(w \times h) \times n$  valeurs aléatoires.

Description de l'architecture de la classe `Random_weights`:

Dans un premier temps, on utilise [3] pour la construction de nombre aléatoires à partir d'une graine (seed) basée sur le temps.

```
1  #include <iostream>
2  #include <chrono>
3  #include <random>
4
5  void Random_weights(double nb_filters, double nb_weights, std::vector<std::vector<double>>&
   nb_tot)
6  {
7      // construct a random generator engine from a time-based seed: Ref[3]
8      unsigned seed = std::chrono::system_clock::now().time_since_epoch().count(); //time;
   system real time
9      std::default_random_engine generator (seed); //random generator engine
10
11     std::normal_distribution<double> distribution (0.0,1.0); //( result_type mean = 0.0,
   result_type stddev = 1.0 )
12 }
13
```

Dans un second temps, on remplit les matrices de filtres (array de type vector). On utilise la ligne de code "*number = distribution(generator)*" dans l'exemple donné dans [4] pour remplir un poid d'un filtre considéré. Ainsi, on ajoute deux boucles à la classe `Random_weights` définies ci-dessous:

```
1  for(int ii = 0; ii < nb_filters; ii++) //loop on the total number of filters
2  {
3      std::vector<double> one_filter; //array initialisation with no defined size
4
5      for (int jj = 0; jj < nb_weights; jj++) //nb_weights = filter height * filter width
6      {
7          double number = (distribution(generator)); //random number from a random generator
   engine, Ref2
8
9          one_filter.push_back(number); // Filling of 1 filter with random values
10
11     }
12
13     nb_tot.push_back(one_filter); //Filling of all filters with random values
14 }
```

On associe à cette classe le code source **Random\_weights.cpp** et l'en-tête **Random\_weights.h**.

**References:**

- [1] "Programmation Generique, Bibliotheque Standard (STL)", Université de Sorbonne, 2018, "<https://www.lpsm.paris/pageperso/roux/enseignements/1819/ifma/chap05.pdf>".
- [2] "std::normal\_distribution", cppreference, 2021, "[https://en.cppreference.com/w/cpp/numeric/random/normal\\_distribution](https://en.cppreference.com/w/cpp/numeric/random/normal_distribution)".
- [3] "std::normal\_distribution::(constructor)", cplusplus, 2021, "[https://www.cplusplus.com/reference/random/normal\\_distribution/normal\\_distribution/](https://www.cplusplus.com/reference/random/normal_distribution/normal_distribution/)".
- [4] "std::normal\_distribution", cplusplus, 2021, "[http://www.cplusplus.com/reference/random/normal\\_distribution/](http://www.cplusplus.com/reference/random/normal_distribution/)".