Introduction de Réseaux de Neurones Profonds

Définition

« Deep learning » est une branche de « Machine learning » basées sur l'apprentissage de modèles de données. Une observation (comme une image) peut être représentée de différentes façons par un vecteur de données en fonction de :

- L'intensité des pixels dont elle est constituée
- Ses différentes arêtes
- Les différentes régions de forme particulière

Définition « Deep » (profondeur)

Profondeur : la longueur de plus long chemin d'entrée à sortie.

La profondeur de **Classique Neuron Network** égale le nombre de couche (e.g. le nombre de couche cachée plus 1 lequel représente la couche sortie). **SVM** (support Vector Machines) a la profondeur 2 (une pour la sortie Kernel ou pour l'espace caractéristique ; une pour la combinaison linéaire générant la sortie.

La profondeur de 2 est suffisante dans de nombreux cas pour approcher n'importe quelle fonction avec une précision arbitraire. Par exemple, pour les portes logiques, les neurones formels (à seuil), les neurones avec une fonction d'activation sigmoïdale, et les unités à base radiale (radial basis fonction, RBF) comme dans les SVMs. Mais cette possibilité a un prix : le nombre de nœuds requis dans le graphe (c'est-à-dire le nombre d'opérations, mais aussi le nombre de paramètres à entraîner) peut devenir très large.

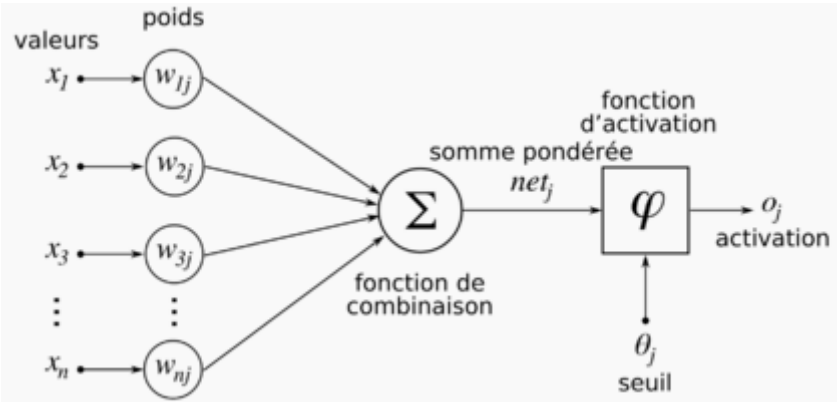
Profondeur des processus cognitifs

- Les humains organisent leurs idées et leurs concepts de façon hiérarchique.
- Les humains apprennent d'abord des concepts simples, et les combinent ensuite pour représenter des concepts plus abstrait.
- Les humains construisent des solutions en combinant de nombreux niveaux d'abstraction et de traitement.

L'introspection des concepts exprimable par le langage suggère aussi une représentation éparse : la description d'une entrée donnée (par exemple, une scène visuelle) fait appel uniquement à une faible fraction de tous les mots et concepts existants.

Rappel le modèle du réseau de neurones

Structure du réseau



Le neurone calcule la somme de ses entrées puis cette valeur passe à travers la fonction d'activation pour produire sa sortie.

Fonction de combinaison

Un neurone reçoit des neurones en amont un certain nombre de valeurs via ses connexions synaptiques, et il produit une certaine valeur en utilisant une fonction de combinaison. Cette fonction peut donc être formalisée comme étant une fonction vecteur-à-scalaire, notamment :

- Les réseaux de type **MLP** (Multi-Layer Perceptron) calculent une combinaison linéaire des entrées, c'est-à-dire que la fonction de combinaison renvoie le produit scalaire entre le vecteur des entrées et le vecteur des poids synaptiques.
- Les réseaux de type **RBF** (Radial Basis Function) calculent la distance entre les entrées, c'est-à-dire que la fonction de combinaison renvoie la norme euclidienne du vecteur issu de la différence vectorielle entre les vecteurs d'entrées.

Fonction d'activation

La fonction d'activation (ou fonction de seuillage/transfert) set à introduire une non-linéarité dans le fonctionnement du neurone.

Les fonctions de seuillage présentent généralement trois intervalles :

1. En dessous du seuil, le neurone est non-actif (souvent dans ce cas, sa sortie vaut 0 ou -1) ;
2. Aux alentours du seuil, une phase de transition ;
3. Au-dessus du seuil, le neurone est actif (souvent dans ce cas, sa sortie vaut 1)

Des classiques de fonctions d'activation

- La fonction sigmoïde

$$f(x) = \frac{1}{1 + e^{-\lambda x}}$$

- La fonction tangente hyperbolique

$$\mathrm{th}(x) = \frac{\mathrm{sh}(x)}{\mathrm{ch}(x)} = \frac{\mathrm{e}^{x} - \mathrm{e}^{-x}}{\mathrm{e}^{x} + \mathrm{e}^{-x}} = \frac{\mathrm{e}^{2x} - 1}{\mathrm{e}^{2x} + 1}$$

- La fonction de Heaviside

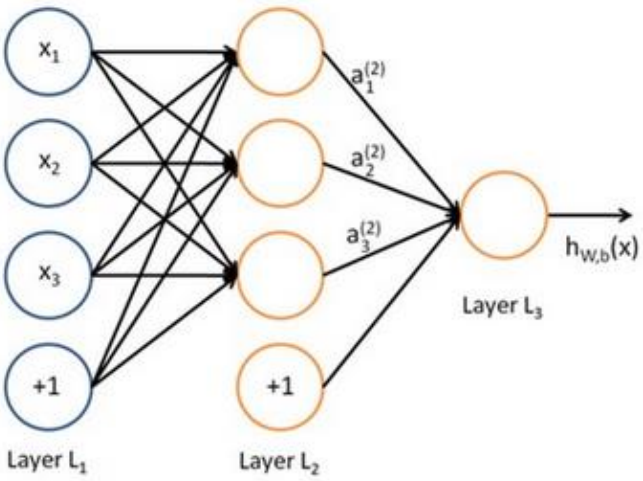
$$\forall x \in \mathbb{R},\ H(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{si} \quad x \geq 0. \end{cases}$$

Propagation de l'information

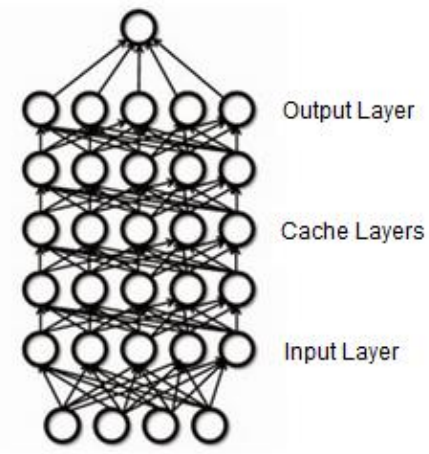
La fonction neuronale est simplement une fonction de seuillage : elle vaut 1 si la somme pondérée dépasse un certain seuil ; 0 sinon. Dans un modèle plus riche, le neurone fonctionne avec des nombres réels (souvent compris dans l'intervalle [0,1] ou [-1,1]). On dit que le réseau de neurones passe d'un état à un autre lorsque tous ses neurones recalculent en parallèle leur état interne, en fonction de leurs entrées.

Deep learning VS classique Neuron Network

Comparaison « Deep learing » à « classique Neuro Network »



Classique Neuro Network



Multip-cache layers Deep learing

Tradition Neuro Network applique « Back Propagation » avec une seule couche cachée : initialisation aléatoire. A partir de l'écart entre le résultat sorti de network et résultat ciblé, il peut changer des paramètres de chaque couche jusqu'à convergence (gradient descendre).

Deep Learning est un entier greedy Layer-wise entrainement de RBM. La stratégie d'entrainement comme suivent :

1. *Phase 1 – non supervisé*. Greedy layer-wise empilage
2. *Phase 2 – supervisé régulier* forward-backward deep learning
3. *Phase 3 – supervisé discriminant* Classification error backpropagation

Nb :

- L'apprentissage non supervisé de représentations est utilisé pour pré-entraîner chaque couche.
- L'apprentissage non supervisé se fait une couche à la fois, chaque couche entraînée après la couche du dessous. La représentation apprise par une couche est utilisée comme entrée par la couche suivante (du dessus).
- L'apprentissage supervisé est utilisé pour raffiner toutes les couches pré-entraînées (ainsi que la couche de sortie, et éventuellement d'autres couches supplémentaires)

Les architectures d'apprentissage en profondeur

Il y a beaucoup de différentes branches de profondeur architecture

Deep neural network

Deep neural network (DNN) est un artificiel neurone network(ANN) avec multiple caches couches entre la couche d'entrée et la couche sortie. Familier à ANNs, DNNs peut être modèle complexe non-linéaire relation.

Un DNN peut être discriminant entrainé par l'algorithme de propagation arrière. Le poids mise à jour via stochastique gradient descendant utilisant l'équation suivant :

$$\Delta w_{ij}(t+1) = \Delta w_{ij}(t) + \eta \frac{\partial C}{\partial w_{ij}}$$

η: le taux d'apprentissage.

C : le coût fonction, qui correspond au type d'apprentissage (non – supervisé, supervisé, renforcement) et la fonction d'activation.

Les problèmes de deep neural network sont sur ajustement et le temps de computation.

Deep belief network

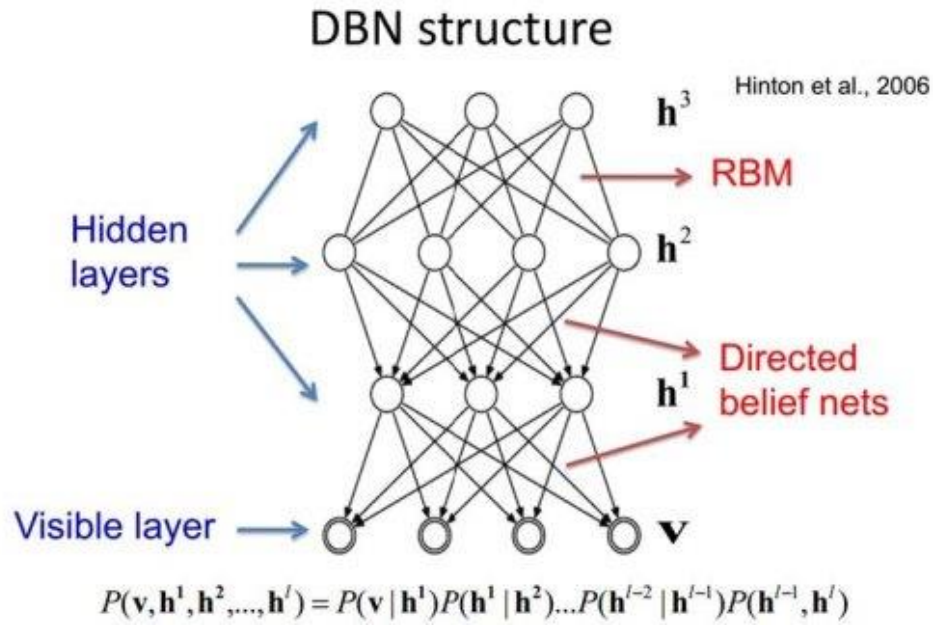
Deep belief network, DBN est un probabiliste, un modèle générative contenant plusieurs couches caches. Il est aussi considéré comme composé de modules d'apprentissage simple.

Le résultat de pré-entrainement DNN peut être utilisé comme les poids initiaux par DBN. En suite DBN peut utiliser propagation arrière ou d'autre algorithme discriminatif pour réglage fin les poids. Cette méthode est vraiment utile quand la base d'apprentissage est limitée, puisque les poids mal initialisés peuvent avoir un impact significatif sur la performance du modèle final. Ces poids sont préformés dans une région de l'espace de poids qui est plus proche des pondérations optimales (comparant à l'initialisation aléatoire). Cela permet à la fois une meilleure capacité de modélisation et de convergence plus rapide de la phase de réglage fin.

DBNs sont des modèles graphiques qui apprennent à extraire une représentation hiérarchique profonde des données d'apprentissage. Ils modélisent la distribution conjointe entre observée vecteur *x* et la l couche cachée h^k :

$$P(x, h^1, \dots, h^l) = \left(\prod_{k=0}^{l-2} P\left(h^k \middle| h^{k+1}\right) \right) P\left(h^{l-1} \middle| h^l\right)$$

Où $x = h^0$, $P\left(h^k \middle| h^{k+1}\right)$ est une distribution conditionnelle pour les unités visible sur les unités cachées de la RBM au niveau k, et $P\left(h^l \middle| h^{l+1}\right)$ est la distribution visible caché conjointe dans le RBM de haut niveau. Ceci est illustré dans la figure ci-dessous.



Un DBN peut être entrainé en la manière de non-supervisé, couche par couche où les couches sont construit par RBM (Restricted Boltzmann machines). Un RBM est un non-orienté, générative basée sur l'énergie avec une couche d'entrée et seule couche cachée. Connexions existent seulement entre

les unités visibles de la couche d'entrée et les unités cachées de la couche cachée. Il n'y a pas de connexions intra la couche entre des nœuds.

La méthode d'entrainement du RBMs est proposée par Geoffrey Hinton pour implémenter avec le modèle « Produit de l'expert » qui est connu comme divergence contrastive (CD). CD fournit une approximation de la méthode du maximum de vraisemblance qui devrait idéalement être appliquée pour l'apprentissage les poids de RBM. Dans l'entraînement d'un seul RBM, mise à jour des poids sont effectuées avec la montée gradient par l'équation suivante :

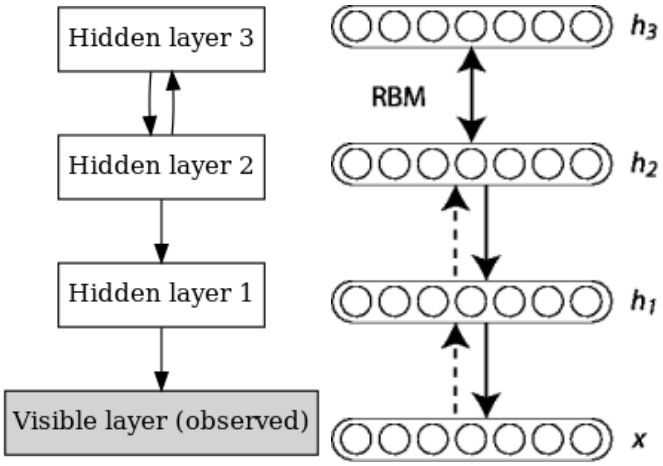
$$\Delta w_{ij}(t+1) = \Delta w_{ij}(t) + \eta \frac{\partial \log(p(v))}{\partial w_{ij}}$$

$p(v) = \frac{1}{Z}\sum_h e^{-E(v,h)}$: La probabilité de vecteur visible. Z est la fonction partition pour normalisation. $E(v,h)$ est l'énergie fonction assigne l'état de network.

Une fois qu'un RBM est formé, un autre RBM peut être empilé au sommet de celui-ci pour créer un modèle multicouche. Chaque fois qu'un autre RBM est empilé, la couche visible d'entrée est initialisée à un vecteur de formation et les valeurs pour les unités dans les couches de RBM déjà formé sont affectées en utilisant les poids et les préjugés actuels. La couche finale des couches déjà formés est utilisée comme entrée à le nouveau RBM. Le nouveau RBM est ensuite formé à la procédure ci-dessus, et alors ce processus peut être répété jusqu'à ce qu'un critère d'arrêt requis soit atteint.

L'apprentissage du Deep Learning

L'algorithme pour RBM

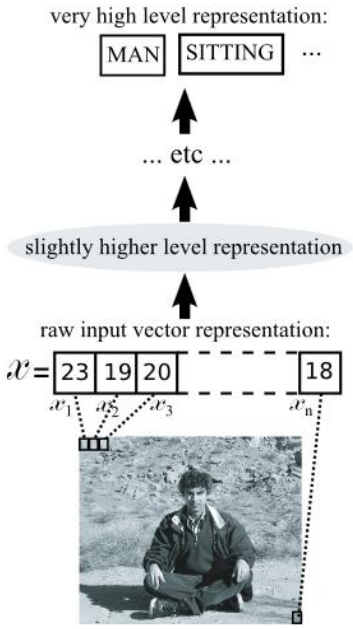


L'entrée matrice X regardant comme une matrice de signatures. Le principal du greedy layer-vise non-supervisé entrainement peut appliquer au DBNs avec RBM autant que les construction blocs pour chaque couche. Le processus comme suivant :

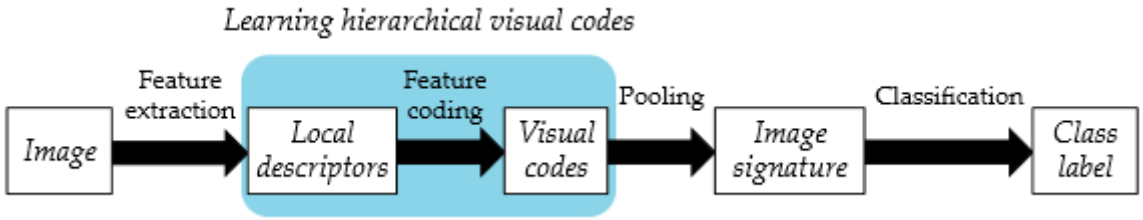
1. Entraîner la première couche comme un RBM, entrée X=h(0), comme la couche visible
2. Utilisant la première couche, on obtient la représentation (la matrice du poids *W*) laquelle sera utilisé comme la base d'entrées pour la deuxième couche. La représentation peut être choisie pour l'activation moyenne $p\left(h^{(1)} = 1 \middle| h^{(0)}\right)$ ou les échantillons de $p\left(h^{(1)} \middle| h^{(0)}\right)$.
3. Entraîner la deuxième couche comme un RBM, utilisant des données transformées (échantillons ou activation moyenne) comme la base d'entraiment (pour la couche visible du RBM)

4. Répéter l'étape 2 et 3 jusqu'à le nombre de couche désireux. Chaque fois il se propage soit des échantillons sois des valeurs moyennes.
5. Affiner tous les paramètres d'architecture profonde adhérant au proxy pour le DBN log (possibilités) ou au critère d'entrainement supervisé (après l'ajout de machines d'apprentissage supplémentaire pour convertir la représentation appris en prédictions supervisés, par exemple un classificateur linéaire)

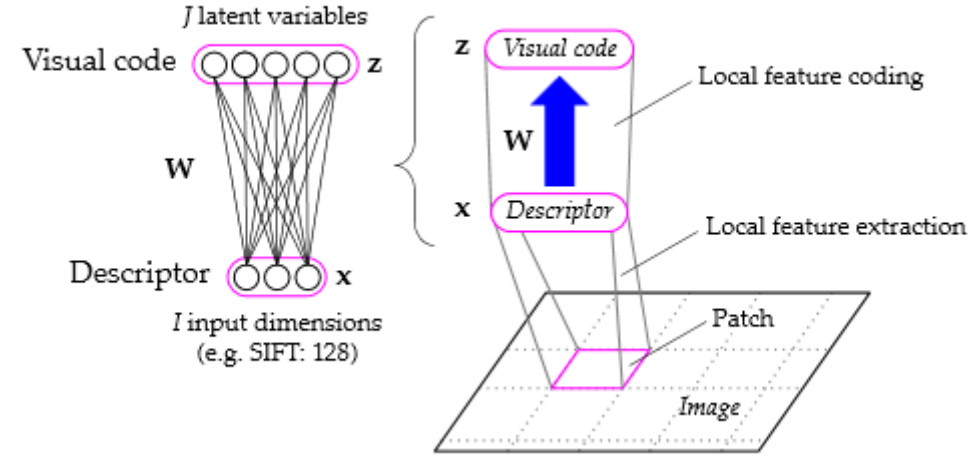
Feature codage processus du Deep learning

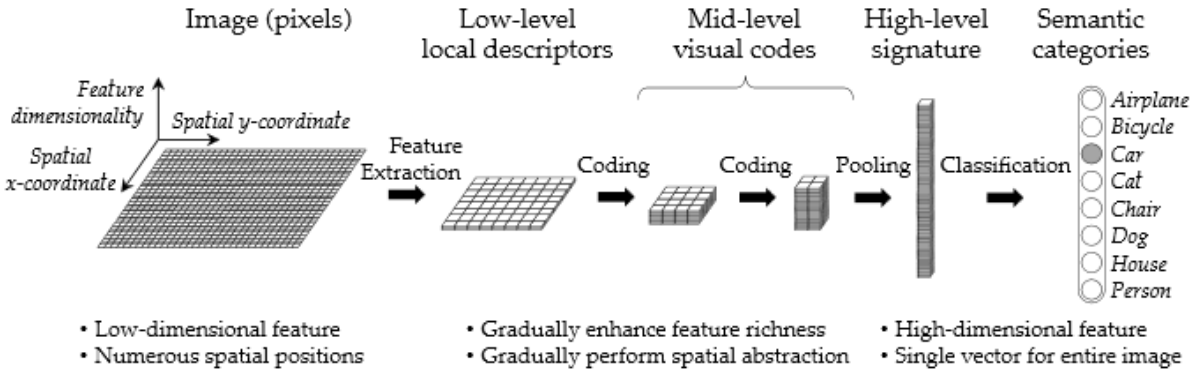


Construire un espace de représentation (feature space)



Descriptor: SIFT (scale invariant feature transform), HOG (histogram of oriented gradients), SURF (speeded-up robust features)





(a) Bag-of-words model with hierarchical feature coding.

Référence :

PhD Thesis: Learning Deep Visual Representations *Hanlin Goh*,

https://tel.archives-ouvertes.fr/tel-00948376

Systèmes OCR

La reconnaissance optique de caractères part de l'image numérique réalisée par un scanner optique d'une page ou un appareil photo numérique et produit en sortie un fichier texte en divers formats (texte simple, formats de traitements de texte, XML…)

Les étapes de traitement peuvent être schématisées ainsi :

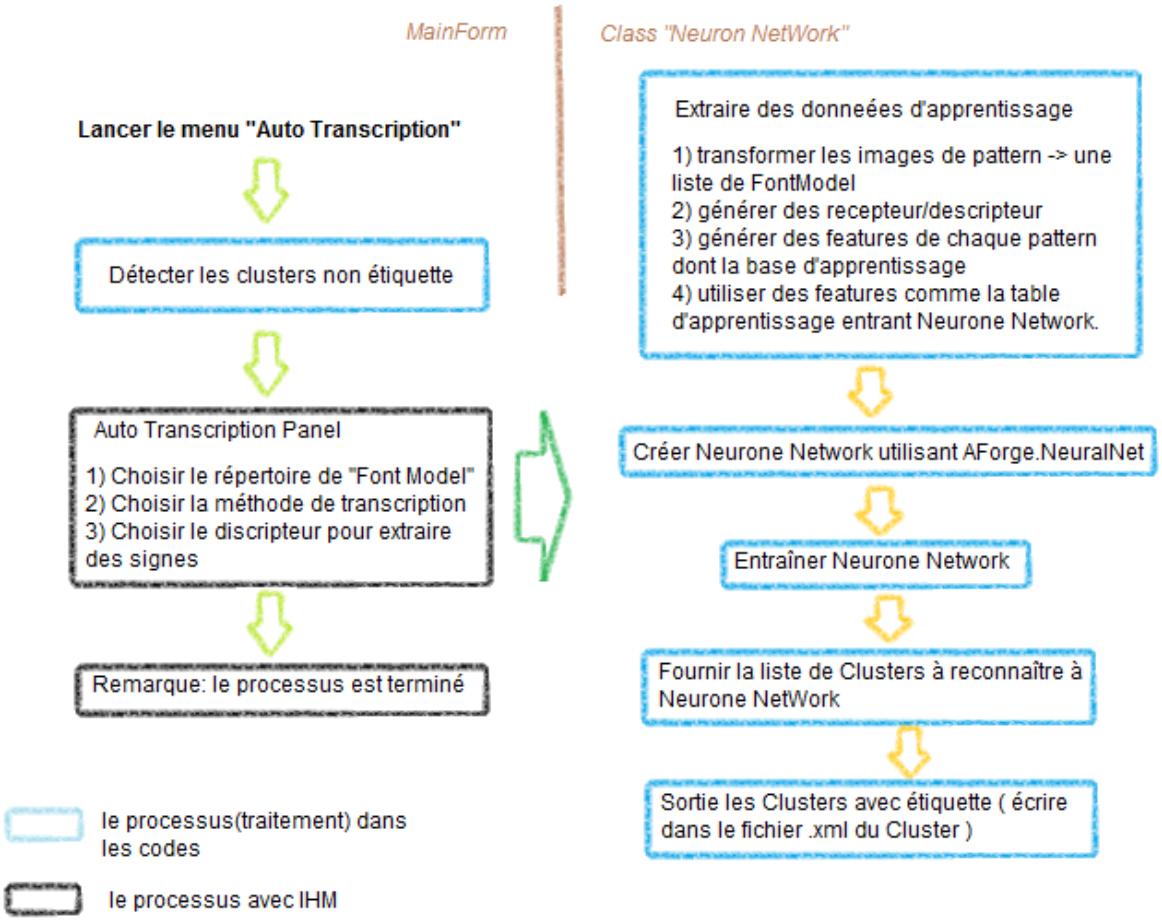
1. **Pré-analyse** de l'image : le but est d'améliorer éventuellement la qualité de l'image. Ceci peut inclure le redressement d'images inclinées, des corrections de contraste, image binaire, la détection de contours.
2. **Segmentation** en lignes et en caractères (ou Analyse de page) : vise à isoler dans l'image les lignes de texte et les caractères à l'intérieur des lignes.
3. **Reconnaissance** : après normalisation, une instance à reconnaitre est comparée à une bibliothèque de formes connues, et on retient la forme la plus « proche » (ou les N formes

les plus proches), selon une distance ou une vraisemblance. Les techniques de reconnaissance se classent en quelques grands types :

1) Classification par Caractéristiques(Features) : une forme à reconnaître est représentée par un vecteur de valeurs numériques calculées à partir de cette forme. Le rôle du classificateur est de déterminer à quelle classe de caractères la forme à reconnaitre appartient le plus vraisemblablement
2) Méthodes métriques : consistent à comparer directement la forme à reconnaître, au moyen d'algorithmes de distance, avec un ensemble de modèles appris.
3) Méthodes statistiques : dans le domaine de la reconnaissance d'écriture manuscrite, il est fréquemment fait appel aux méthodes probabilistes/statistiques comme les chaînes de Markov

4. **Post-traitement** utilisant des méthodes linguistiques et contextuelles pour réduire le nombre d'erreurs de reconnaissance : systèmes à base de règles, ou méthodes statistiques basées sur des dictionnaires de mots, de syllabes.
5. Génération du format de sortie, avec la mise en page pour les meilleurs systèmes

Auto Transcription dans Retro2014



Référence logiciel :

- Deep Belif Network and Boltzmann Machines: http://accord-framework.net/samples.html
- Neural Network OCR: http://www.codeproject.com/Articles/11285/Neural-Network-OCR

Quel type de neural network vas-tu utiliser ? Quelles classes/méthodes de Aforge.NET Neural Network ?
Quelle structure de réseau ? Nb de neurones/couche ? nb de couches ?
Quelles caractéristiques en entrée du réseau (features,signatures) ?
Quelles données d'apprentissage vas-tu utiliser pour apprendre les paramètres du réseau ?

Model de Neuron Network: Deep Belief Network, RBM (Restricted Boltzmann machines)

Constructeur du network:

DeepBeliefNetwork(int inputsCount, params RestrictedBoltzmannMachine[] layers)

RestrictedBoltzmannMachine(IStochasticFunction function, int inputsCount, int hiddenNeurons)

Libraries:
Accord.Neuro.Networks.DeepBeliefNetwork, Accord.Neuro.Networks.RestrictedBoltzmannMachine

Nb de neurones/couche :

RestrictedBoltzmannMachine : neurone cachée- 10

Nb de couches :

DeepBeliefNetwork :4 couches - 1 couche entrée, 2 couches caches(RBM), 1 couche sortie

Matrice d'entrée : signatures de « ZernikeDescriptor »

Type d'apprentissage : supervisé

Données d'apprentissage : R80_Gering_cicero_v2

OCR : Optical character recongnition

DBN : Deep Belif Network

RBM : Radial Boltzmann machines

MLP : Multi-Layer Perceptron

ANN : Artificiel Neurone Network