# Photo-z estimation by Deep learning

*«Aproach by images»*

**Présenté par :** *Ousmane Issa*
**Sous la direction de :** *Pr. Engelbert Mephu Nguifo*

# Table des matières

**I Photo-z estimation by Deep learning *«Aproach by images»* 3**

**1 Travaux existants dans la littérature 7**
1.1 Méthodes basées sur les modèles (template-based) . . . . . 7
1.2 Méthodes empiriques . . . . . 8

**2 Modèles de l'apprentissage automatique 10**
2.1 Réseau de neurones artificiels . . . . . 10
2.1.1 Perception multi-couches (MLP) . . . . . 11
2.1.2 Auto-Encoder . . . . . 12
2.2 Réseau de neurones convolutifs (CNN) . . . . . 13
2.2.1 Couche convolutive . . . . . 15
2.2.2 Couche d'agrégation (pooling) . . . . . 18
2.3 Boosting . . . . . 18
2.4 Apprentissage profond . . . . . 19
2.4.1 Apprentissage par rétro-propagation . . . . . 19

**3 Résultats et discussion 21**
3.1 Données . . . . . 21
3.1.1 Source et nature de données . . . . . 21
3.1.2 Pré-traitement sur les images . . . . . 22
3.2 Configurations techniques . . . . . 23
3.3 Configuration des modèle . . . . . 23
3.3.1 CNN (type : inception module) . . . . . 23
3.3.2 Boosting (type : xgboost) . . . . . 23
3.4 Expérimentions avec CNN . . . . . 24
3.5 Expérimentions avec XGBoost . . . . . 24
3.6 Discussion . . . . . 24

## *Résumé :*

## *Abstract :*

# Introduction

Après avoir introduit, le contexte du problème est détaillé, il est suivi par la motivation.

# Première partie

# Photo-z estimation by Deep learning

*«Aproach by images»*

## Décalage vers le rouge (Redshift)

Le redshift est l'augmentation de la longueur d'onde d'un objet lors de l'application de radiation électromagnétique sur cet objet. Un observateur peut remarquer le déplacement de la lumière de cet objet vers le rouge, ce qui est un effet contraire au blueshift (décalage vers le bleu) où l'observateur remarque le déplacement de la lumière de l'objet vers le bleu.

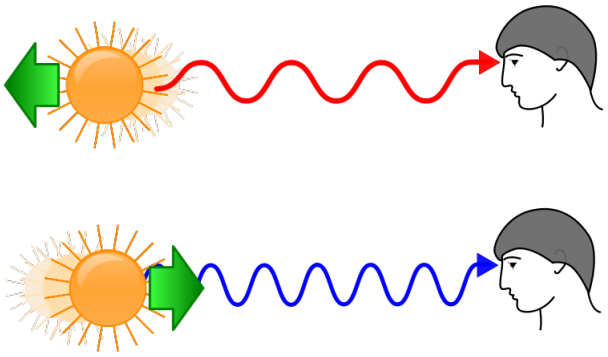

FIGURE 1 – illustration du redshift et blueshift

Le reshift est noté $z$ et se manifeste quand l'objet en mouvement tout en s'éloignant de l'observateur (effet dopler) ou il est du à l'expansion de l'univers (redshift cosmologique) ou due à la gravitation (redshift gravitationel). Dans notre cas le redshit qu'on essaie d'estimer correspond au redshift cosmologique et il s'obtient de la manière suivante : $1 + z = \frac{\lambda_{observ}}{\lambda_{emis}}$ où $\lambda_{observ}$ et $\lambda_{observ}$ sont respectivement la longueur d'onde observée et celle émise. La loi de Huble établit une relation entre la distance et le redshift cosmologique comme suit $v = H_0 d = cz$ où $d$ est la distance, $v$ la vitesse, $c$ la vitesse de la lumière, $H_0$ la constance de Huble et $z$ le redshift cosmologique.

## Spectroscopique

La méthode spectroscopique mesure le redshift par la dispersion de la lumière en comparant le décalage entre les lignes spectrales des éléments chimiques émises et celles absorbées. La spectroscopie donne des estimations précises du redshift, néanmoins elle est couteuse et prend beaucoup de temps. Ce qui encourage les scientifiques à chercher d'autres moyens moins couteux pour estimer le redshift.

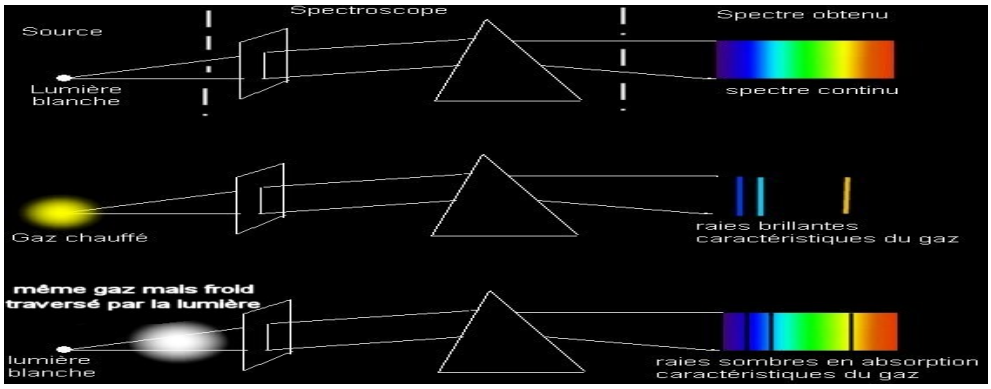


FIGURE 2 – Expériences spectroscopiques

## Estimation par photométrie

La méthode basée sur la photométrie utilise un ensemble de filtres profonds dans des longueurs d'ondes spécifiques (5 bandes dans notre cas) pour capturer certaines caractéristiques de l'objet concerné. Puisque c'est une opération rapide, la photométrie donne une grande quantité d'observations en terme de nombre et de profondeurs de caractéristiques de chacune des observations. À partir des caractéristiques de l'objet observé, on tente de déterminer son décalage vers le rouge. Donc la précision du calcul du redshift par données photométries dépend de la profondeur du mesure de ces caractéristiques. SDSS a collecté plus de 300 millions des objets photométries. Le télescope LSST (Large Scale Survey Telescope) qui sera bientôt déployé donne plus de caractéristiques que ceux existants (utilise 6 bandes allant de l'ultra violet à l'infra-rouge), qui va rendre plus précis l'estimation du redshift par photométrie. Dans ce travail nous essayons d'estimer directement le redshift à partir des images.

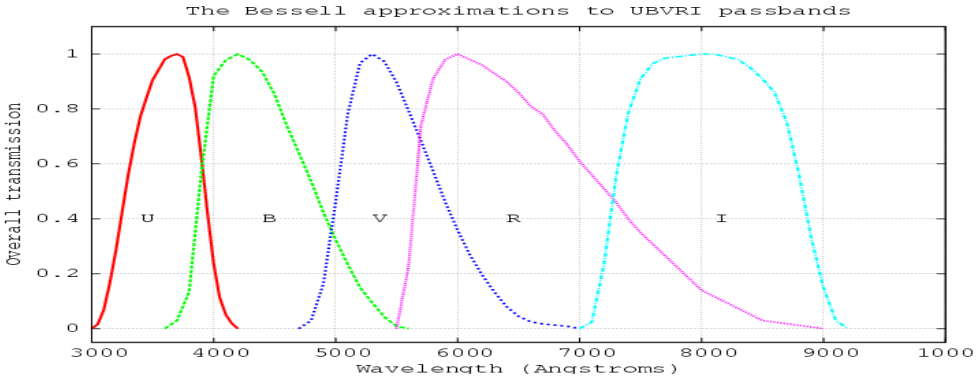


FIGURE 3 – Expériences photométries

Ces derniers années plusieurs travaux [12, 9, 6, 17, 26] ont été fait à partir des données photométries pour l'estimation du redshift. La méthode d'estimation du redshift par données photométries est divisée en deux sous méthodes à savoir celle basée modèle et celle basée sur les techniques de l'apprentissage automatique et statistiques dite empirique. Ces deux sous-méthodes seront détaillées dans le chapitre 1.

## Qualité de prédiction du redshift

Il existe plusieurs paramètres pour définir la qualité de prédiction du redshift, ils sont tous basé sur une fonction de perte normalisée (erreur normalisée). L'objectif de ce travail est de trouver une fonction ($f$) qui prend en entrée des images superposées dans 5 bandes de longueurs d'onde différentes d'un corps cosmologique et retourne le redshift correspondant. Pour cela une fonction ($\Delta z$) dite fonction d'erreur est définie, elle permet de vérifier si la fonction $f$ prédit bien en générale le redshift d'un corps en faisant la soustraction entre la valeur exacte du redshift (valeur spectroscopique) et celle prédite(estimée) $\Delta z = z_{phot} - z_{spec}$.
L'erreur normalisée est définie comme suit :
$\Delta z_{norm} = \frac{\Delta z}{z_{spec}+1}$
Dans ce travail nous utilisons principalement quatre paramètres de mesure de qualité de prédiction du redshift :

- Le biais qui est la moyenne de l'ensemble des erreurs noté $\mu(\Delta z_{norm})$ pour l'erreur normalisée.
  $\mu(\Delta z_{norm}) = \frac{\sum_{i=1}^{n} \Delta z_{norm_i}}{n}$
- Le standard déviation noté $\sigma(\Delta z_{norm}) = \sqrt{\frac{\sum_{i=1}^{n} (\Delta z_{norm_i} - \mu(\Delta z_{norm}))^2}{n}}$
- La proportionnalité des outliers/particuliers catastrophiques mesurée par $(\frac{card(|\Delta z_norm| \geq 0.15)}{n}) * 100$ ou $(\frac{card(|\Delta z_norm| \geq 3\sigma(\Delta z_norm))}{n}) * 100$ en pourcentage.
- Erreur moyenne quadratique (RMSE) qui est similaire à la déviation standard. $RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\Delta z_{norm_i})^2}{n}}$.
  Où $n$ est le nombre total de l'échantillon de test, et $card$ est la fonction cardinale.

# Chapitre 1

# Travaux existants dans la littérature

## 1.1 Méthodes basées sur les modèles (template-based)

Ces méthodes sont basées sur de modèles prédéfinis utilisant la distribution spectral de l'énergie. Elles supposent un ensemble de modèles de galaxies et tire le modèle le plus adapté et ensuite deduire le redshift [2, 12, 17].
Cette méthode de mesure du redshift est biaisée (seulement les galaxies) et ainsi entraîne des erreurs dans la prédictions des erreurs. À partir de template-based on peut déterminer d'autres composants physiques que le redshift [9]. Plusieurs tentatives de l'amélioration de la précision du redshift de galaxies ont été faites, ci-dessous un tableau des certains travaux intéressants de la littérature.

| paper | database | type/SED TEMP | $\mu(Z_{norm})$ | $std(Z_{norm})$ | $\|Z_norm\| \geq 0.15(\%)$ |
|---|---|---|---|---|---|
| [31] | PHAT | GALAXY/LP Flat Err | 0.0013 | 0.0445 | 9.88 |
| | CAPR | Galaxy/LP HDF Err | -0.0081 | 0.0490 | 7.65 |
| [2] | HDF | GALAXY/BPZ | 0.08 | | |
| | | Galaxy/ML | 0.10 | | |
| [1] | SDSS DR12 | GALAXY | 0.0000584 | 0.0205 | 4.11 |

TABLE 1.1 – Résultat concernant la méthode basée modèle

Dans [2] deux modèles probabilistes ont été utilisés : maximum likelihood (ML) et le modèle bayesian (BPZ).

Dans [31], l'analyse est faite directement sur les différentes bases de données sans les déplacer. Cela permet non seulement d'éviter un télé-chargement long des données et l'obligation d'avoir un très grand espace de stockage, mais également de profiter de la bonne structuration de bases de données relationnelles. Ils ont également fait l'analyse sur multi-bandes variés, ce qui ne permet de faire

que la méthode d'analyse basée modèle. Deux types de modèles de distribution l'énergie spectrale (SED) de galaxies ont été utilisés (BPZ et LP). Le modèle Bayesian a été sélectionné pour l'analyse.

[1] est une méthode hybride, utilisant en même temps la régression (méthode empirique) et l'adaptation à un modèle de galaxies (méthode basée modèle). Cette méthode bat toutes les autres mais elle est biaisée car elle s'est focalisée uniquement sur les galaxies.

## 1.2 Méthodes empiriques

Les méthodes empiriques tentent de réduire l'erreur de prédiction du redshift par les méthodes statistiques et machines learning. Ces méthodes ont besoin des données étiquetées (chacun de vecteurs dans l'ensemble d'apprentissage doit être étiqueté par son redshift). Ce qui oblige à fusionner les données de la photométrie et spectroscopiques des mêmes objets (dans catalog [1], les objets avec les mêmes objid).La proportionnalité des images étiquetées par leur redshift est très petite par rapport à celles non étiquetées.

| paper | database | type/method | $\mu(Z_{norm})$ | $std(Z_{norm})$ | $\|Z_norm\| \geq 0.15(\%)$ |
|---|---|---|---|---|---|
| [26] | SDSS DR12 | Galaxy/AdaBoost stacking | 0.0008 ($\mu^{50}$) | 0.0248 ($\sigma_{68}$) | 0.733 |
| [14] | SDSS DR 10 | Galaxy/DNNs | 0.00 | 0.03 ($\sigma_{68}$) | 1.71 |
| | | AdaBoost | $-0.001$ | 0.03 ($\sigma_{68}$) | 1.56 |
| [9] | SDSS DR 13 | All type/DNN | -0.0255009 | 0.152743 | 9.83078 |
| | | All type/XGB | 0.00149561 | 0.168866 | 10.7685 |
| [6] | SDSS DR9 | All type/DCMDN | -0.007 | 0.014 | |

TABLE 1.2 – Résultats concernant la méthode empirique

Dans [5], le réseau de neurones artificiel a été testé sur les données photométries sous forme de vecteurs (magnitudes et couleurs) de type galaxy provenant de SDSS et obtenir une précision rms (root mean square) de 0.0238.
Dans le papier [26] plusieurs modèles et architectures ont été testé et le meilleur parmi est le AdaBoost empilé. AdaBoost fait partir de la classe des algorithmes d'apprentissage de boosting qui fait l'apprentissage sur plusieurs modèles hierarchiquement en augmentant à chaque nouveau la distribution des éléments mal ajustés. Mais dans [26] on fait en plus du boosting mais également un empilement de modèles renforcés. Ces modèles renforcés sont hiérarchisés et chaque modèle suivant tient compte du redshift prédit du precedant.
Ben Hoyle, dans [14] utilise les réseaux de neurones profonds et AdaBoost pour estimer le redshift à partir des images provenant de la collecte SDSS DR 10. Dans [9], XGBoost et MLP a été utilisé pour estimer le redshift sans pré-classification (Galaxies, étoiles et quasars confondus)et diminuer ainsi le biais d'estimation. [6] utilise la mixture de densité gaussian et le réseau convolutif. À partir des réseaux convolutifs, un ensemble des paramètres de la distribution gaussian sont

1. http ://skyserver.sdss.org/dr13/en/help/browser/browser.aspx

estimés, ainsi il obtient la meilleure deviation (0.014) standard et une erreur moyenne de -0.007 de l'estimation du redshift.

# Chapitre 2

# Modèles de l'apprentissage automatique

Ce chapitre est consacré aux différents modèles d'apprentissage profonds. Il se focalisera principalement sur les réseaux de neurones convolutifs puisqu'ils ont plus été utilisé dans ce travail que le reste, et il se terminera par un petit détail sur un récent modèle basé sur les arbres de décision.

## 2.1 Réseau de neurones artificiels

Le réseau de neurones artificiels est une fonction mathématique s'inspirant du réseau de neurones biologique. Il prend en entrée un vecteur de dimension d qu'il va par la suite transformer en un autre vecteur de dimension k en sortie. Pendant cette transformation, chaque élément de position $i$ du vecteur d'entrée est multiplié par un réel $w_{ij}$ pour donner le vecteur $S_j$. L'élément j du vecteur de la sortie est obtenu en faisant la somme des éléments du vecteur $S_j$. Pour avoir une forme non linéaire, une fonction non linéaire est appliquée sur la sortie. Les $w_{ij}$ sont appelés les poids de connexions entre les neurones. Chaque vecteur est appelé une couche, ainsi le vecteur d'entrée est appelé la couche d'entrée et celui de sortie la couche de sortie. Une couche intermédiaire est appelée une couche cachée.
Soit $I = (I_1, ..., I_d)$ le vecteur d'entrée et $S = (S_1, ..., S_k)$ celui de la sortie, $W$ la matrice des poids et $A$ la fonction d'activation. $\forall j = 1..k\ S_j = A(\sum_{i=1}^{d} W_{ij} * I_i)$.

L'apprentissage est l'étape de calibrage des paramètres, nous avons trois modes d'apprentissage : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement. Notre travail est basé sur l'apprentissage supervisé. L'apprentissage supervisé suppose la connaissance au préalable de l'étiquette (valeur à estimer pour une entrée) de chaque individu de l'ensemble de l'échantillon d'entrée.À partir de cet échantillon étiqueté, un algorithme d'apprentissage essaie de trouver une corrélation entre les données et leurs étiquettes

en donnant des valeurs aux poids du réseau. Quant à l'apprentissage non supervisé, avec des données non étiquetées l'algorithme essaie de trouver la corrélation.

### 2.1.1 Perception multi-couches (MLP)

— **Perception**

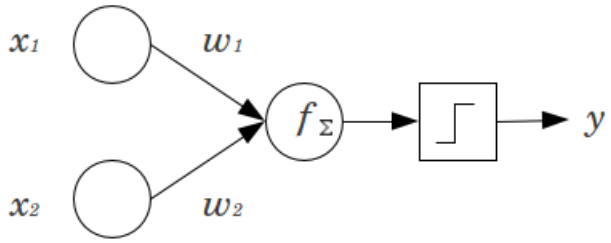


FIGURE 2.1 – Perceptron

Le perception est un réseau de neurone artificiel à une couche d'entrée et une de sortie sans couche cachée. Comme on peut voir un exemple sur la figure 2.1, le perception est le modèle de réseau de neurones le plus simple. Son défaut est son incapacité à estimer une fonction non linéaire.

— **Perception multicouches (MLP)**

Le MLP(Multi-layer perceptron) est un réseau de neurones à au moins une couche cachée. On peut le considérer comme une séquence d'un ensemble de perceptrons. Cette manière de sequencer les perceptrons et d'activer chaque sortie de chaque percetron par une fonction non linéaire emerge un comportement non linéaire permettant ainsi d'estimer les fonctions non linéaire. Le nombre de séquence de perceptron défini la profondeur de l'architecture du réseau. Le MLP augmente le nombre de paramètres(poids) à calibrer, en fonction de l'augmentation de la profondeur. L'augmentation de la profondeur peut facilement amener le modèle à spécialiser le calibrage des poids sur les données d'entrée (sur-apprentissage), ce qui entraîne une mauvaise estimation d'un nouveau exemple.

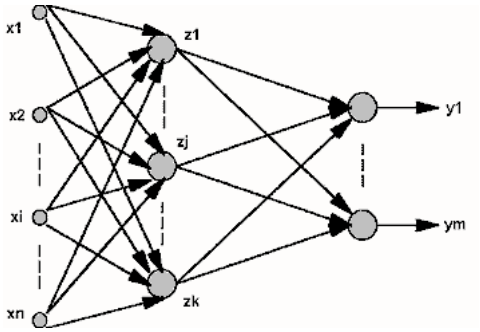


FIGURE 2.2 – MLP

### 2.1.2 Auto-Encoder

Un auto-encoder est un réseau de neurones dont sa phase d'apprentissage est non supervisée et chaque sortie du réseau est semblable à l'entrée. Le vecteur résultant est l'une des couches cachée appelée le code qui est un condensé de la représentation de l'entrée, il est comme l'analyse de la composante principale (PCA) mais il permet de plus une représentation non linéaire par rapport au PCA [**autoencoder**]. La partie de l'architecture réseau avant l'obtention du code est appelé codeur et celui après le décodeur, comme on le voir sur la figure 2.3.

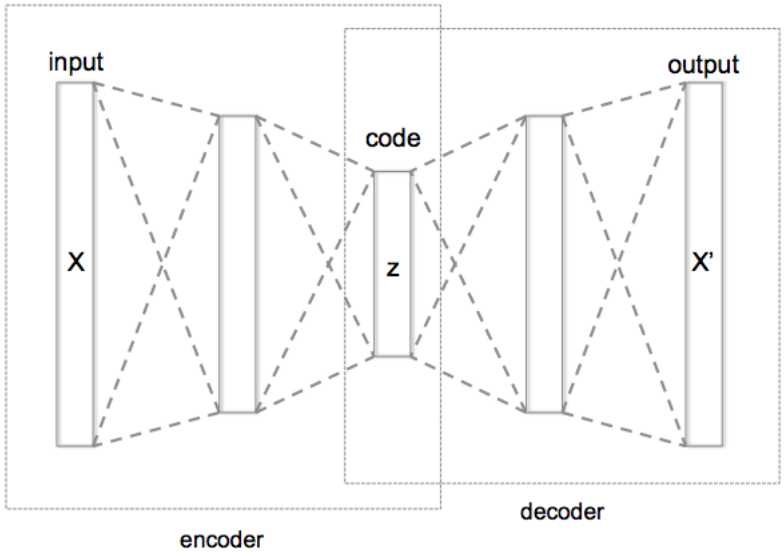


FIGURE 2.3 – Auto-encoder

## 2.2 Réseau de neurones convolutifs (CNN)

Les réseaux de neurones convolutifs sont de type de réseau de neurone travaillant sur des images en se focalisant sur des caractéristiques locaux de l'image qu'il transforme d'une abstraction à une autre à travers les couches []. Il s'inspire du cortex visuel du chat, ce qui le rend plus approprié pour la vision artificielle.

Le réseau convolutif permet non seulement l'obtention d'une bonne précision par rapport au réseau de neurone classique mais aussi permet la diminution du nombre de paramètre du réseau à apprendre (les poids) à travers le partage des poids.

**Noyau ou filtre(Kernel), Opération de convolution, Déplacement (strides), Padding** []
Le noyau est la matrice de paramètres qui se connecte sur la partie locale de l'image pour l'opération de convolution, donc cette partie locale doit avoir la même dimension que le noyau. Le noyau contient alors les paramètres (poids) à apprendre et il sera utilisé pour tout le reste des parties locales de même taille de l'image en faisant passer une fenêtre de déplacement sur l'image. L'opération de convolution est de multiplier case par case deux matrices pour obtenir une matrice M, puis faire la somme des éléments de M pour une obtenir un sca-

laire, c'est donc une fonction non inversible. Le déplacement est le nombre de case pour se déplacer vers la droite et en bas pour positionner le noyau sur une nouvelle partie locale de l'image pour une autre opération de convolution. Au coin supérieur gauche de l'image la position du déplacement est (0,0), puisque une opération complète de convolution donne comme résultat une image intermédiaire A qui est une abstraction de l'ancienne, la valeur de A(i,j) correspond au résultat de l'opération de convolution à la position du déplacement (i,j).À la position (p, q) de déplacement, la prochaine position à droite est (p+1, q) et en bas (p, q+1). On remarque rapidement la diminution de la dimension de l'image , le padding permet d'augmenter la taille de l'image par des zéros pour avoir la même dimension de sortie que l'entrée.

**Fonctionnement**

Le réseau convolutif divise la matrice de l'image en des petites matrices(de même taille que le noyau) ordonnées par leurs positions et ces petites matrices passe dans le même réseau de neurones (c'est-à-dire les mêmes poids : noyau) pour donner une nouvelle matrice en sortie qui sera à son tour une entrée d'une nouvelle couche et ainsi de suite jusqu'à la sortie. Dans mon explication ci-dessus certes j'ai fait plus d'abstraction de certains détails, il y a pas que de couches de convolution qui existe, il y à des couches de regroupement (pooling), de couches d'activation (ReLu (généralement utilisé), softMax, tanh, ...), de couche de simplification pour empecher de passer en surapprentissage (dropout, batch normalization, ...) []

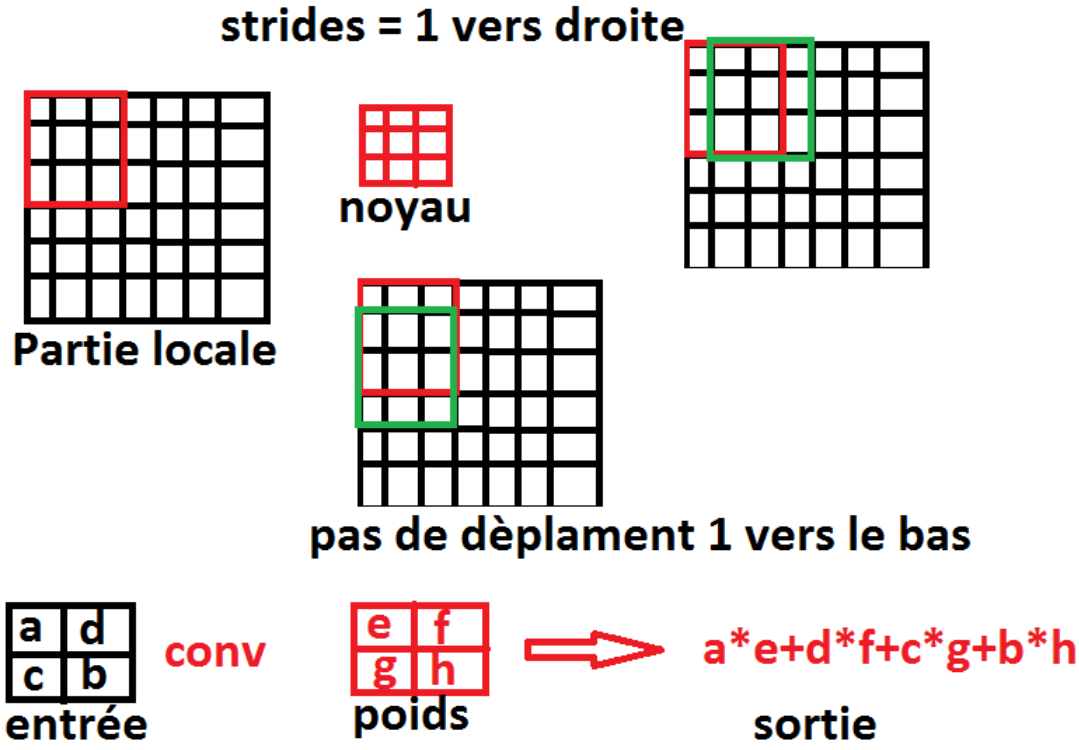


FIGURE 2.4 – fonctionnement du réseau de neurone convolutif

### 2.2.1 Couche convolutive

Plusieurs amélioration de la couche convolutive ont été proposées dans la littérature et ont donné des bons résultats surtout sur la base de données de Image-Net []Nous avons plusieurs de réseau convolutifs, mais nous avons présenter que quelques en détail.

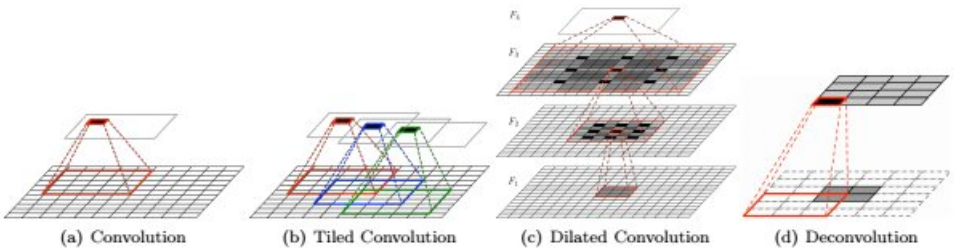


FIGURE 2.5 – Illustration de type quelques de CNN

### Tiled convolutional network (TCN)

Le réseau convolutif d'origine n'utilise qu'un seul noyau qui est partagé entre toutes les parties locales de l'image. Il permet d'éviter des problèmes de translation mais pas d'autres dérangement dans l'image telle la rotation. Le TCN permet d'utiliser à chaque couche convolutive plusieurs noyaux, ainsi permettant plus d'abstractions[23, 36, 43].

### Transposed convolution ou Deconvolution (TC)

Une couche convolutive de CNN prend une image (matrice) en entrée et retourne une autre image de dimension inférieur ou égale à l'image d'origine, le TC fait l'opération inverse. Le TC transforme chaque case de la matrice de l'image en une petite matrice. Ce n'est pas vraiment une opération inverse du CNN comme l'apparence nous fait croire mais c'est juste le type d'entrées et sorties sont inversé. Pour un CNN, les poids se fixent sur une localité et retourne un scalaire tandis pour TC on part d'un scalaire pour arriver à une matrice locale. Pour plus de détail on peut voir [40, 42, 41, 21, 35]

### Dilated convolutional neural

Le dilated convolution neural élargit de plus le filtre en mettant autant des zeros que nécessaires (un hyper-parametre à donner) entre une case et ses voisines de la matrice noyau.Pour plus de detail concernant le dilated convolutional [38, 18, 24, 30].

### Network in network (NiN)

[20] Le NiN est une des améliorations la plus intéressante par rapport aux précédentes, elle permet dans chaque couche convolutive d'utiliser comme filtre un petit réseau de neurones profonds. C'est-à-dire sur la partie locale, est appliquée un petit réseau de neurones profond au lieu d'un perceptron comme dans le cas du CNN. C'est qui veut dire qu'une couche de NiN est contituée d'une couche de convolution de noyau (k, m) suivi par plusieurs couches de convolution de noyau (1,1) chacune.

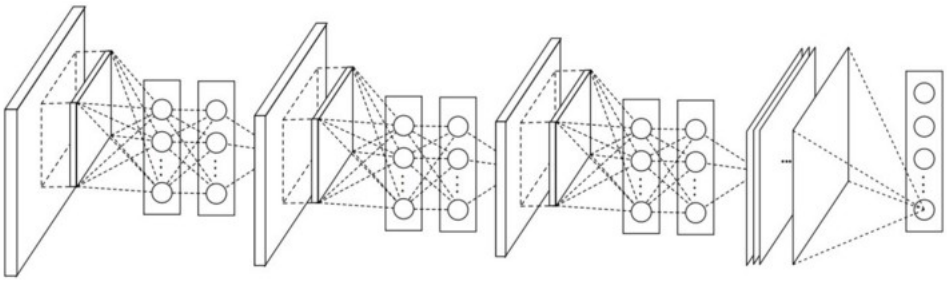

FIGURE 2.6 – Illustration NiN

### Inception module

[33, 13, 34, 32] Inception module est une généralisation de NiN, dans chacune de ses couches on peut avoir plusieurs successions de couches convolutives de noyau de dimension ($k_i$,$m_i$) où les ($k_i$,$m_i$) ne sont pas nécessairement des couples de 1, les resultats de cet ensemble de couches successives sont fusionnés pour donner un ensemble d'images qui sont plus d'abstraction (profondeur). Un type de ce modèle a gagné le prix du challenge de la reconnaissance visuelle sur ImageNet en 2015 (GoogleNet [33]). Plusieurs amélioration de ce modèle ont été fait, il y a eu même une fusion avec le residual network qui donne plus de précision et permet de réduire le temps d'apprentissage également [33, 13, 34, 32]. Notre modèle s'inspire de ce type de réseau convolutif.

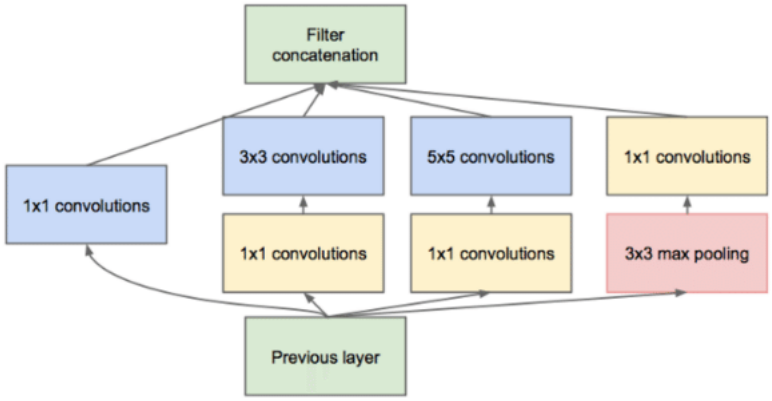


FIGURE 2.7 – Illustration Inception module

### 2.2.2 Couche d'agrégation (pooling)

La couche d'agrégation permet non seulement la réduction de la dimension mais également permet de capturer certaines abstractions statistiques locales [15, 37, 39, 29]. Plusieurs couches d'agrégation ont été définies parmi elles la couche maxpooling et celle de averagePooling sont les plus utilisées. La couche maxpooling permet de choisir la valeur maximale locale parmi les valeurs de la région considérée et le averagepooling fait la moyenne de la région locale considérée.

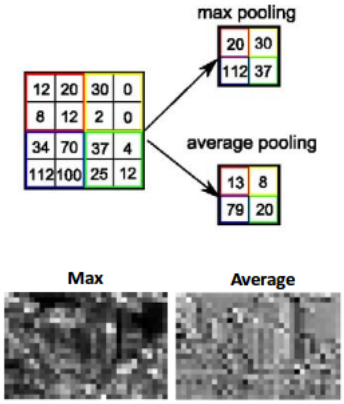


FIGURE 2.8 – maxpooling et averagepooling

## 2.3 Boosting

Le boosting est une technique d'apprentissage qui s'appuie sur des modèles légers appris pour construire de modèles sophistiqués. Il opère de manière séquentielle d'un modèle à un autre tout en corrigeant l'erreur du précèdent en augmentant la distribution de ceux qui sont mal prédits. L'ensemble des modèles légers n'est pas nécessairement homogène, on peut faire le boosting avec plusieurs modèles de familles différentes. Ce type de modèle ne cessent de nous surprendre dans certains domaine par rapport aux réseaux de neurones profonds comme dans notre étude actuelle.

**Regularized gradient boosting tree**

Un arbre de décision est une fonction/modèle permet d'estimer une valeur ou classifier en construisant un arbre dont les feuilles sont les classes ou les valeurs estimées. Formellement un arbre de décision est décrit comme suit :
Soit $x \in R^n$ l'objet pour lequel une valeur y doit être prédite, $f$ la fonction correspondant à l'arbre de décision. $f(x) = W_{q(x)}$ où $W \in R^d$ est le vecteur correspondant aux valeurs predites et $q : R^n -> \{1..d\}$.

gradient boosting tree est la version du boosting où le modèle léger est un arbre de décision et l'ensemble de ces modèles est obtenu en minimisant le gradient, du passage d'un modèle à un autre. Pour éviter le problème du

sur apprentissage, les techniques de régularisation sont appliquées. Supposons que nous avons $k$ estimateurs (modèles légers appris), l'estimation d'une valeur correspondant à $x$ est $\widehat{y} = \sum_{i=1}^{k} f_i$

Le problème d'optimisation se résume à minimiser l'expression suivante : $L(\Phi) = \sum_{i=1}^{n} l(y, \widehat{y}) + \sum_{j=1}^{k} \Omega(f_j)$ où $l(y, \widehat{y})$ est la fonction de perte, $\Omega(f_j)$ permet de contrôler la complexité du modèle pour empêcher le sur-apprentissage et $\Phi$ est l'ensemble de paramètres. $\Omega(f_j) = \frac{1}{2}\lambda \sum_{i=1}^{m} W_i^2 + \alpha \sum_{j=1}^{m} |W_j| + \gamma m$, où $\lambda$, $\alpha$ et $\gamma$ sont des hyper-paramètres.
Son implantation est xbboost, pour plus de details [1]

## 2.4 Apprentissage profond

Le problème d'apprentissage se résume à un problème d'optimisation, çà revient à adapter les paramètres du modèle pour une génération du monde concerné (population) à partir d'un échantillon des données. L'apprentissage profond est l'adaptions des paramètres d'un modèle construit à partir de la superposition de plusieurs couches de réseau de neurones. La Superposition des couches permet de capturer plus d'abstractions. Ces paramètres sont appelés les poids (weights) du réseau.

### 2.4.1 Apprentissage par rétro-propagation

La rétro-propagation permet de propager l'erreur en arrière sur toutes les couches pour ajuster les poids.

#### Optimisation

Trouver les poids adaptés se ramène à optimiser une certaine fonction. Cette fonction dépend du type d'architecture de réseaux de neurones. La prédiction d'un individu X par une architecture de n couches en mode en feed foward (comme MLP) est $\widehat{Y} = Y_n$ tel que $Y_1 = \sigma(W_1 * X + b_1)$, $Y_2 = \sigma(W_2 * Y_1 + b_2)$, $...Y_n = \sigma(W_(n-1) * X + b_(n-1))$ où $W_i$ sont les poids du réseaux, $b_i$ les biais et $\sigma$ la fonction d'activation [..., 28, 22, 8]. Pour l'obtention des poids adaptés d'une architecture feed forward, il revient à minimiser la fonction $\sum_{i=1}^{m} L(Y^i, \widehat{Y}^i)$ où $Y^i$ est la vraie valeur correspondante à X, m est la taille de l'échantillon et $L$ est la fonction de perte ainsi dans notre cas $L(Y, \widehat{Y}^i) = (\frac{Y - \widehat{Y}}{Y+1})^2$. Pour la minimisation, la descente du gradient est utilisée, pour chaque étape du gradient les paramètres et biais sont mis en jours. $W^i = W^{i-1} - \alpha \frac{\delta L}{\delta W} W^{i-1}$ où $\alpha$ est le taux d'apprentissage .

1. http ://xgboost.readthedocs.io.

Le problème majeur du gradient est la difficulté d'obtention du minimum global, la rapidité de convergence et son coup de calcul. Plusieurs améliorations du gradient ont été effectuées concernant ces problèmes (SGD [11], AdaGrad [7], Adam [19], ...). Pour ce travail Adam et SGD ont été utilisés.

**Régularisation**

La régularisation permet de contrôler la complexité du modèle en pénalisant la fonction à optimiser d'une certaine valeur. Ce qui permet au modèle de généraliser mieux le monde concerné et éviter le sur apprentissage. Plusieurs méthodes de régularisations existent : **Batch normalisation [16]** qui normalise les données à l'entrée de la couche suivant la couche du batch, **L1 et L2** (qui augmente sur la fonction de parametres à optimiser la quantité respectivement $\lambda * \sum_{i=1}^{m} |w_i|$ et $\lambda * \sqrt{\sum_{i=1}^{m} w_i^2}$) où $\lambda$ est un hyper-paramètre, et dropout qui supprime certains neurones d'une certaine proportionnalité $\alpha$ de la couche concernée. Dans ce travail batch normalisation, L2 et dropout ont été utilisés.

# Chapitre 3

# Résultats et discussion

## 3.1 Données

Dans ce chapitre, nous allons décrire les pré-traitement sur les données (images) et mentionner leur source ainsi que le mode d'accès.

### 3.1.1 Source et nature de données

#### SDSS(Sloan Digital Sky Survey IV)

En termes simples, le Sloan Digital Sky Survey est l'étude astronomique la plus ambitieuse jamais entreprise. SDSS a observé en détail un quart de l'ensemble du ciel, déterminant les positions et la luminosité absolue de centaines de millions d'objets célestes[4]. Il mesure également les distances à plus d'un million de galaxies et quasars [1].

#### DR13 (database release version 13)

[2] Data Release 13 stocke les prèmieres données de la quatièmes phase de la collecte de SDSS. Elle inclut les données prises à partir de 25 juin 2015, and englobe plus que un tiers de la sphère céleste. Avec plusieurs mesures du ciel faites et dans plusieurs façons la taille de la base de données DR12 peut de plusieurs manières, 1231051050 objets (catalogues) ont été collectés [3]

1. http ://skyserver.sdss.org/dr13/en/sdss/sdsshome.aspx
2. http ://skyserver.sdss.org/dr13/en/help/browser/browser.aspx
3. http ://www.sdss.org/dr13/scope/

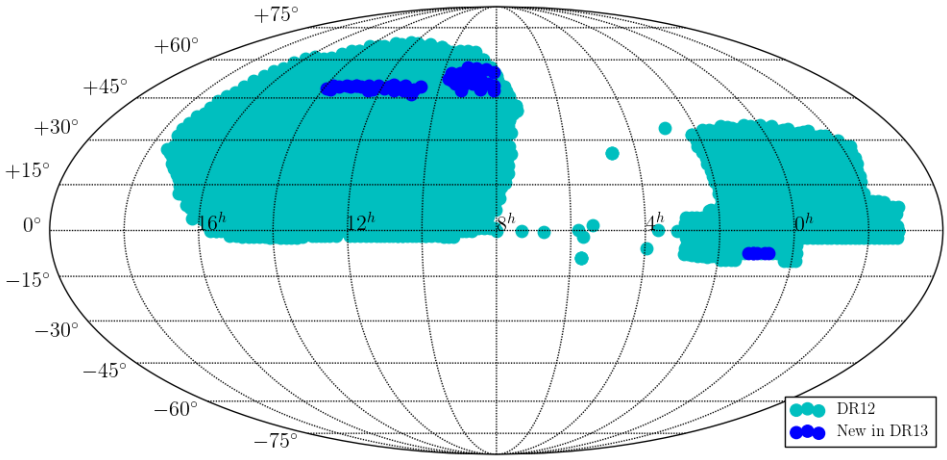


FIGURE 3.1 – eBOSS surveys

Toutes nos données proviennent de la collecte de données SDSS . Elles sont télé-chargées sous forme des vecteurs chacune dans 5 bandes différentes (griuz). À partir de ces données sous forme de vecteurs provenant des tables SpecPhoto et PhotoPrimary de DR13 de SDSS, les urls des images correspondant à chaque vecteur sont générés. À partir de ces urls, les images sont télé-chargées et sauvegardées localement.

### 3.1.2 Pré-traitement sur les images

Les images dans leur taille d'origine sont très grandes (1361*2048 pixels), et contiennent des quantités lumineuses très variées. La taille de chaque image est réduite à 32*32, cette réduite est faite en utilisant les coordonnées centrales (rowc_ et colc_) des objets(images) dans chaque bande. Pour normaliser, le minimum de la quantité lumineuse (pixel) de chaque image est soustraite de l'image et le logarithme est appliqué à chaque pixel de l'image.

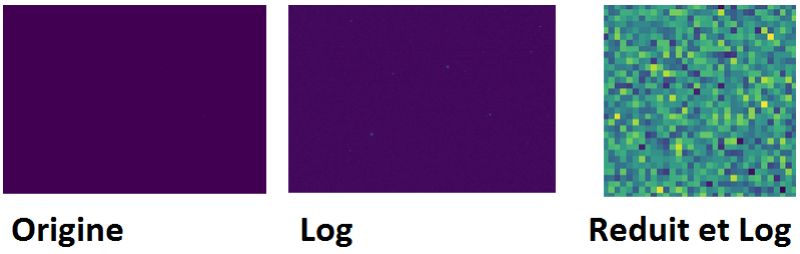


FIGURE 3.2 – pretement sur les images

## 3.2 Configurations techniques

Pendant toutes nos expérimentations, nous avons utilisés des machines hpc (haute performance computing) équipées de processeurs graphiques. Elles sont du type

Pour la concrétisation de notre architecture du CNN, nous avons utilisé la bibliothèque Keras [3] et pour celle du boosting nous avons utilisé la bibliothèque skLearn [25].

## 3.3 Configuration des modèle

Dans ce travail, nous avons choisi d'utiliser deux principales modèles de l'apprentissage automatique à savoir boosting qui est un modèle ensembliste, et réseau de neurones convolutifs (apprentissage profonds). En effets le boosting et les réseaux convolutifs ont donnée des bons résultats non pas seulement sur l'estimation du redshift [9, 6] mais également sur d'autres problèmes de régression et classification [10, 27].

Le choix et l'adaptation des hyper-paramètres ont été fait manuellement pour les deux types de modèles en faisant des test sur un petit échantillon de données et quelques combinaisons des hyper-paramètres. Au niveau du CNN, l'obtention des bons hyper-paramètres est compliquée car le CNN possède beaucoup d'hyper-paramètres à adapter.

### 3.3.1 CNN (type : inception module)

Notre modèle final, celui qui s'adapte bien à nos données c'est-à-dire qui réduit mieux l'erreur normalisée ($\Delta z_{norm}$) est un réseau convolutif de la famille des inceptions modules.

### 3.3.2 Boosting (type : xgboost)

Trois principaux hyper-paramètres (nombre d'estimateurs, taux d'apprentissage et la profondeur de l'arbre) ont été réglés et adapté et tout le reste des paramètres ont été laissé avec leurs valeurs par défaut. Les trois modèles configurés du boosting sont mentionnés dans le tableau 3.1.
**Quelques paramètres commun** learning_rate = 0.1, colsample_bylevel=0.592, reg_alpha=0.651, reg_lambda=2.84, seed=5555

| modèles | n_estimators | max_depth |
|---|---|---|
| model1 | 150 | 6 |
| model3 | 150 | 5 |
| model2 | 200 | 5 |

TABLE 3.1 – Paramètres de différents modèles de xgboost

3.4 Expérimentions avec CNN

3.5 Expérimentions avec XGBoost

3.6 Discussion

# Conclusion et perspectifs

# Appendix

**For Galaxy**

SELECT TOP 200000 p.objid,s.rowc_u,s.rowc_i,s.rowc_z,s.rowc_g,s.rowc_r,s.colc_u,s.colc_i,s.colc_z,s.colc
as redshift,p.class FROM SpecPhoto AS p JOIN PhotoPrimary AS s ON s.objid
= p.objid where p.class = 'GALAXIES'

**For Quazars**

SELECT TOP 200000 p.objid,s.rowc_u,s.rowc_i,s.rowc_z,s.rowc_g,s.rowc_r,s.colc_u,s.colc_i,s.colc_z,s.colc
as redshift,p.class FROM SpecPhoto AS p JOIN PhotoPrimary AS s ON s.objid
= p.objid where p.class = 'QSO'

**For stars**

SELECT TOP 200000 p.objid,s.rowc_u,s.rowc_i,s.rowc_z,s.rowc_g,s.rowc_r,s.colc_u,s.colc_i,s.colc_z,s.colc
as redshift,p.class FROM SpecPhoto AS p JOIN PhotoPrimary AS s ON s.objid
= p.objid where p.class = 'STAR'

# Bibliographie


[1] R. Beck et al. « Photometric redshifts for the SDSS Data Release 12 ». In : (2016). MNRAS, arXiv :1603.09708v2.

[2] Narciso. Benitez. « Bayesian photometric redshift estimation ». In : *arXiv :astro-ph/9811189v1* (1998). doi : 10.1086/308947.

[3] François Chollet et al. *Keras*. https://keras.io. 2015.

[4] François Chollet et al. *SDSS Data Release 13*. http://www.sdss.org/dr13/.

[5] ADRIAN A. COLLISTER et OFER LAHAV. « ANNZ : ESTIMATING PHOTOMETRIC REDSHIFTS USING ARTIFICIAL NEURAL NETWORKS ». In : *arXiv :astro-ph/0311058v2* (2004). doi : 10.1086/383254.

[6] A. D'Isanto et K.L. Polsterer. « Photometric redshift estimation via deep learning ». In : *Astrophysics - Instrumentation and Methods for Astrophysics, arXiv :1706.02467 [astro-ph.IM]* (2017). doi : 10.1051/0004-6361/201731326.

[7] John Duchi, Elad Hazan et Yoram Singer. « Adaptive Subgradient Methods for Online Learning and Stochastic Optimization ». In : *Journal of Machine Learning Research* (2010).

[8] « Empirical evaluation of rectified activations in convolutional network ». In : *Proceedings of the International Conference on Machine Learning (ICML) Workshop* (2015).

[9] Rim Shayakhmetov et Engelbert Mephu Nguifo. « Deep Learning for Photo-z estimation ». In : *Data Science Innovative Days MADICS 2017, poster session,* (2017). doi : github.com/shayakhmetov/redshift_prediction/blob/master/Poster_MaDICS.pdf.

[10] « Experimenting XGBoost Algorithm for Prediction and Classification of Different Datasets ». In : *National Conference on Recent Innovations in Software Engineering and Computer Technologies (NCRISECT)* (2017).

[11] Matteo Fischett, Iacopo Mandatelli et Domenico Salvagnin. « Faster SGD training by minibatch persistency ». In : (2018). arXiv :1806.07353v1.

[12] Pieter G. van Dokkum Gabriel B. Brammer et Paolo Coppi. « EAZY : A FAST, PUBLIC PHOTOMETRIC REDSHIFT CODE ». In : *Astrophysical Journal* (2007).

[13] K. He et al. « Deep residual learning for image recognition ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). pp. 770–778.

[14] Ben Hoyle. « Measuring photometric redshifts using galaxy images and Deep Neural Networks ». In : *arXiv :1504.07255v2 [astro-ph.IM]* (2016).

[15] A. Hyvärinen et U. Köster. « Complex cell pooling and the statistics of natural images ». In : *Network : Computation in Neural Systems* (2007). pp. 81–100.

[16] Sergey Ioffe et Christian Szegedy. « Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift ». In : *arXiv :1502.03167v3 [cs.LG]* (2015).

[17] M.Bolzonella J.-M.Miralles et R. Pello. « Photometric redshift based on standard SED fitting procedures ». In : *A&A 363, 476* (2000). doi : arXiv:astro-ph/0003380.

[18] N. Kalchbrenner et al. « Neural machine translation in linear time ». In : (). CoRR abs/1610.10099.

[19] Diederik P. Kingma et Jimmy Lei Ba. « A Method for Stochastic Optimization ». In : (2017). arXiv :1412.6980v9.

[20] M. Lin, Q. Chen et S. Yan. « Network in network ». In : *Proceedings of the International Conference on Learning Representations (ICLR)* (2014).

[21] J. Long, E. Shelhamer et T. Darrell. « Fully convolutional networks for semantic segmentation ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2017). pp. 640–651.

[22] A.L. Maas, A.Y. Hannun et A.Y. Ng. « Rectifier nonlinearities improve neural network acoustic models ». In : *Proceedings of the International Conference on Machine Learning (ICML), 30* (2013).

[23] J. Ngiam et al. « Tiled convolutional neural networks ». In : *Proceedings of the Advances in Neural Information Processing Systems (NIPS)* (2010). pp. 1279–1287.

[24] A. v. d. Oord et al. « Wavenet : A generative model for raw audio ». In : (). CoRR abs/1609.03499.

[25] F. Pedregosa et al. « Scikit-learn : Machine Learning in Python ». In : *Journal of Machine Learning Research* 12 (2011), p. 2825-2830.

[26] Roman.Zitlau Ben.Hoyle Kerstin.Paech Jochen.Weller Markus Michael Rau et Stella Seitz. « Stacking for machine learning redshifts applied to SDSS galaxies ». In : *arXiv :1602.06294v2 [astro-ph.IM]* (2016). doi : 10.1093/mnras/stw1454.

[27] « Recent Advances in Convolutional Neural Networks ». In : (2017). arXiv :1512.07108v6.

[28] « Rectified linear units improve restricted Boltzmann machines ». In : *Proceedings of the International Conference on Machine Learning (ICML)* (2010).

[29] O. Rippel, J. Snoek et R. P. Adams. « Spectral representations for convolutional neural networks ». In : *Proceedings of the Advances in Neural Information Processing Systems (NIPS)* (2015). pp. 2449–2457.

[30] T. Sercu et V. Goel. « Dense prediction on sequences with time-dilated convolutions for speech recognition ». In : *Proceedings of the Advances in Neural Information Processing Systems (NIPS) Workshops* (2016).

[31] Robert.Beck Laszlo.Dobosa Tamas.Budavari Alexander S.Szalay et Istvan Csabaia. « Photo-z-SQL : integrated,exible photometric redshift computation in a database ». In : *arXiv :1611.01560v2 [astro-ph.GA]* (2017). doi : 10.1016/j.ascom.2017.03.002.

[32] C. Szegedy, S. Ioffe et V. Vanhoucke. « Inception-v4, inception-resnet and the impact of residual connections on learning ». In : *Proceedings of the AAAI Conference on Artificial Intelligence* (2017). pp. 4278–4284.

[33] C. Szegedy et al. « Going deeper with convolutions ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015). pp. 1–9.

[34] C. Szegedy et al. « Rethinking the inception architecture for computer vision ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). pp. 2818–2826.

[35] F. Visin et al. « Reseg : A recurrent neural network for object segmentation ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)Workshops* (2015).

[36] Z. Wang et T. Oates. « Encoding time series as images for visual inspection and classification using tiled convolutional neural networks ». In : *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Workshops* (2015).

[37] D. Yu et al. « Mixed pooling for convolutional neural networks ». In : *Proceedings of the Rough Sets and Knowledge Technology (RSKT)* (2014). pp. 364–375.

[38] F. Yu et V. Koltun. « Multi-scale context aggregation by dilated convolutions ». In : *Proceedings of the International Conference on Learning Representations (ICLR)* (2016).

[39] M. D. Zeiler et R. Fergus. « Stochastic pooling for regularization of deep convolutional neural networks ». In : *Proceedings of the International Conference on Learning Representations (ICLR)* (2013).

[40] M. D. Zeiler et R. Fergus. « Visualizing and understanding convolutional networks ». In : *Proceedings of the European Conference on Computer Vision (ECCV)* (2014). pp. 818–833.

[41] M. D. Zeiler, G. W. Taylor et R. Fergus. « Adaptive deconvolutional networks for mid and high level feature learning ». In : *Proceedings of the International Conference on Computer Vision (ICCV)* (2011). pp. 2018–2025.

[42] M. D. Zeiler et al. « Deconvolutional networks ». In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010). pp. 2528–2535.

[43] Y. Zheng et al. « Time series classification using multi-channels deep convolutional neural networks ». In : *Proceedings of the International Conference on Web-Age Information Management (WAIM)* (2014). pp.298–310.