

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Maxime PORTAZ

Thèse co-dirigée par **Philippe MULHEM, Chercheur, CNRS et**
par Jean-Pierre CHEVALLET, MCF, UGA

préparée au sein du **Laboratoire d'Informatique de**
Grenoble
dans l'**École Doctorale Mathématiques, Sciences et**
technologies de l'information, Informatique

Accès à de l'information en mobilité par l'image pour la visite de Musées

**Réseaux profonds pour l'identification de
gestes et d'objets**

Thèse soutenue publiquement le **24 octobre 2018**,
devant le jury composé de :

Denis Pellerin

Professeur à l'Université Grenoble Alpes, Président

Véronique Eglin

Professeur à l'INSA de Lyon, Rapporteur

Lynda Tamine-Lechani

Professeur à l'Université Paul Sabatier de Toulouse, Rapporteur

Hervé Glotin

Professeur à l'Université de Toulon, Examinateur

Philippe Mulhem

Chercheur CNRS, directeur de thèse

Jean-Pierre Chevallet

Maître de conférence à l'Université Grenoble Alpes, directeur de thèse



*Accès à de l'information en mobilité par
l'image pour la visite de Musées - Réseaux
profonds pour l'identification de gestes et
d'objets*

THÈSE PRÉSENTÉE ET SOUTENUE
PAR
MAXIME PORTAZ
EN VUE DE L'OBTENTION DU TITRE DE
DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

LABORATOIRE D'INFORMATIQUE DE GRENOBLE

UNIVERSITÉ GRENOBLE ALPES
GRENOBLE, FRANCE
SEPTEMBRE 2018

© 2018 - MAXIME PORTAZ
TOUT DROIT RESERVÉ.

Accès à de l'information en mobilité par l'image pour la visite de Musées - Réseaux profonds pour l'identification de gestes et d'objets

RÉSUMÉ

Cette thèse aborde le problème d'accès à l'information en mobilité. On s'intéresse à comment rendre l'information à propos des œuvres accessible automatiquement aux visiteurs de lieux touristiques. Elle s'inscrit dans le cadre du projet GUIMUTEIC, qui vise à équiper les visiteurs de musées d'un outil d'aide à l'accès à l'information en mobilité. Être capable de déterminer si le visiteur désire avoir accès à l'information signifie identifier le contexte autour de lui, afin de fournir une réponse adaptée, et réagir à ses actions.

Ce travail est lié aux problématiques d'identification de points d'intérêts, pour déterminer le contexte, et d'identification de gestes des utilisateurs, pour répondre à leurs demandes. Dans le cadre de notre projet, le visiteur est donc équipé d'une caméra embarquée. L'objectif est de fournir une solution à l'aide à la visite, en développant des méthodes de vision pour l'identification d'objet, et de détection de gestes dans les vidéos à la première personne.

Nous proposons dans cette thèse une étude de la faisabilité et de l'intérêt de l'aide à la visite, ainsi que de la pertinence des gestes dans le cadre de l'interaction avec un système embarqué. Nous définissons une nouvelle approche pour l'identification d'objets grâce à des réseaux de neurones profonds siamois pour l'apprentissage de similarité entre les images, avec apprentissage des régions d'intérêt dans l'image. Nous explorons également l'utilisation de réseaux à taille réduite pour la détection de gestes en mobilité. Nous présentons pour cela une architecture utilisant de nouveaux types de bloc de convolutions, pour réduire le nombre de paramètres du réseau et permettre son utilisation sur processeur mobile. Pour évaluer nos propositions, nous nous appuyons sur plusieurs corpus de recherche d'image et de gestes, créés spécialement pour le projet.

*Information Access in mobile environment for museum visits -
Deep Neural Networks for Instance and Gesture Recognition*

ABSTRACT

This thesis is part of the GUIMUTEIC project, which aim is to equip museum tourist with an audio-guide enhanced by a camera. This thesis address the problem of information access in mobile environment, by automatically providing information about museum artefacts. To be able to give this information, we need to know when the visitor desire guidance, and what he is looking at, to give the correct response.

This raises issues of identification of points of interest, to determine the context, and identification of user gestures, to meet his demands. As part of our project, the visitor is equipped with an embedded camera. The goal is to provide a solution to help with the visit, developing vision methods for object identification, and gesture detection in first-person videos.

We propose in this thesis a study of the feasibility and the interest of the assistance to the visit, as well as the relevance of the gestures in the context of the interaction with an embedded system. We propose a new approach for objects identification thanks to siamese neural networks to learn images similarity and define regions of interest. We are also exploring the use of small networks for gesture recognition in mobility. We present for this an architecture using new types of convolution blocks, to reduce the number of parameters of the network and allow its use on mobile processor. To evaluate our proposals, we rely on several corpus of image search and gestures, specifically designed to match the constraints of the project.

CETTE THÈSE EST DÉDIÉE À MES PARENTS.

Remerciements

Après avoir passé trois ans au Laboratoire d’Informatique de Grenoble, pour la réalisation de mon travail de thèse, j’aimerais remercier toutes les personnes qui m’ont aidé dans la réalisation de ce projet. Je tiens tout d’abord à remercier mes directeurs de thèse, Philippe Mulhem et Jean-Pierre Chevallet. Nous avons passé trois années à travailler ensemble avec un très bonne entente, et une ambiance chaleureuse. Les longs débats enrichissants et les discussions plus décontractées passés ensemble ont été un plaisir. C’est grâce à eux que j’ai pu réaliser cette thèse dans les meilleures conditions, grâce à leur aide, leur support et leur amitié.

Je remercie en particulier les professeurs Véronique Eglin et Lynda Tamine-Lechani d’avoir consacré du temps à lire et à rapporter mon travail de thèse. Leur travail de lecture approfondie de ce manuscrit a permis une analyse détaillée et des commentaires enrichissants sur ce travail. J’aimerais également exprimer ma gratitude aux professeurs Hervé Glotin et Denis Pellerin pour avoir accepter d’examiner mon travail et de participer à ma soutenance.

Je voudrais également remercier tous les membres de l’équipe MRIM avec lesquels j’ai pu travailler, directement ou non. Georges Quénot, Catherine Bérut et Lorraine Goeriot ont participé à rendre l’équipe vivante et intéressante, d’un point de vue scientifique et sociale. Les doctorants de l’équipe MRIM, Anuvabh Dutt, Jibril Frej, Nawal Ould-Amer et Seydoux Doubia, ont été très présents avec moi et les discussions que nous avons pu avoir ont été d’une grande aide dans la réalisation de ce travail. Les personnes que j’ai encadré, Matthias Kohl et Marion Schmitt, ont apportés à l’équipe et à moi-même de nouvelles idées, du dynamisme, et je les en remercie.

Les laboratoire d’informatique de Grenoble à été un endroit agréable où travailler, et pour cela je remercie tous les doctorants et ingénieurs avec qui j’ai pu partager repas, pauses café, soirées, et toutes les activités sociales qui ont rendu cette expérience très agréable. Je remercie particulièrement : Sami Alkhoury, Belen Baez, Anil Goyal, Vera Shalaeva, Karim

Assaad, Alexandre Berard, Carole Plasson, Carole Adam, Tien Nguyen, Emeric Grange et Baptiste Vernier. Je remercie Dominique Vaufreydaz, qui a été un ami, un collègue, et une personne sur qui j'ai pu compter depuis que j'ai commencé l'informatique à Grenoble et qui m'a donné le goût de la recherche et de l'enseignement.

Les personnes qui ont rendu tout cela possible, sont, bien entendu, ma famille. Mes parents m'ont soutenu et encouragé. Ma compagne, Radhia, a été un support psychologique et scientifique indispensable de chaque instant. Je remercie toute ma famille, Maud, Lilian, Jamila et Ilyes.

Table des matières

1	INTRODUCTION	3
1.1	Le Contexte de cette thèse	3
1.2	Problématiques de la recherche d'information muséale	4
1.3	Contributions	5
1.4	Plan du manuscrit	7
2	LES ENJEUX DE L'INTERACTION DANS LES VISITES TOURISTIQUES	9
2.1	Les nouveaux moyens d'interaction et d'accès à l'information muséale	9
2.2	Le système GUIMUTEIC	12
3	ÉTAT DE L'ART	17
3.1	Réseaux de neurones profonds	18
3.2	Réseaux de neurones convolutifs	29
3.3	Réduction de la taille et de la complexité des réseaux	35
3.4	Réseaux avec détection de régions	40
3.5	Identification d'instances	43
3.6	Détection d'événement dans les vidéos	51
3.7	Conclusion	53
4	IDENTIFICATION D'ŒUVRES	55
4.1	Similarité entre les images	55
4.2	Représentation d'image et similarité	57
4.3	Plongement des images	58
4.4	Réseau siamois	58
4.5	Choix des triplets	60
4.6	Évaluation	62

5	DETECTION DE RÉGIONS D'INTÉRÊTS	69
5.1	Carte d'activation	69
5.2	Apprentissage sur différentes régions	72
5.3	Apprentissage de régions et de similarité	73
5.4	Expérimentation	75
5.5	Indexation et augmentation de données côté base de données	77
6	DÉTECTION DE GESTES	81
6.1	Utilisation des convolutions groupées pour réduire la complexité	82
6.2	Fusion d'information de plusieurs trames de la vidéo	90
6.3	Conclusion et limitations	96
7	CONCLUSION	99
A	CORPUS POUR L'ÉVALUER DES MÉTHODES DE RECHERCHES D'INFORMATION MULTIMÉDIA	103
A.1	Corpus d'images pour la recherche d'instance dans un musée d'art	103
A.2	Corpus de recherche d'image et de vidéo dans un musée d'héritage culturel	104
A.3	Garofou_I	104
A.4	Garofou_V	105
A.5	Analyse de l'interaction à base de geste dans le cadre de visites culturelles	108
B	ETUDES DE LA PERTINENCE DE L'INTERACTION À BASE DE GESTE	113
B.1	Le dispositif proposé	113
B.2	Les participants	114
B.3	Séances participatives	114
B.4	Bilan	116
C	RÉSEAU DE RECONNAISSANCE DE GESTES	119
C.1	Implémentation	119
C.2	Fusion des informations	121
	RÉFÉRENCES	138

Table des figures

1.1	Exemple d'une aide à la visite avec un smartphone. L'utilisateur peut prendre en photo une œuvre pour avoir des informations sur celle-ci.	4
2.1	Exemple d'un système basé sur l'interaction à base de geste dans un musée [1]	12
2.2	Les 5 gestes mis en avant par les séances de conceptions participative.	13
2.3	Schéma d'organisation du système d'information GUIMUTEIC.	15
3.1	Schéma d'un neurone avec trois connexions entrantes.	19
3.2	Réseaux de neurones à 1 et 2 couches.	20
3.3	Exemple d'apprentissage d'un réseau de neurone.	21
3.4	Fonctions d'activation non-linéaires.	25
3.5	Illustration du <i>dropout</i> pris depuis [2]	27
3.6	Illustration d'une convolution sur une matrice $4 * 4$	29
3.7	Application d'une convolution $3x3$ sur une matrice de taille $4x4$ avec un padding de 1 et un pas de 1	30
3.8	Couche de convolution avec 3 canaux en entrée et 5 en sortie.	31
3.9	Illustration des opérations de convolution, de ReLU et de <i>pooling</i>	31
3.10	Schéma du réseau profond LeNet [3]	32
3.11	Architecture AlexNet [4]	32
3.12	Architecture VGG [5]	33
3.13	Exemple d'un bloc résiduel de ResNet (source [6]).	35
3.14	Bloc Inception (source [7]).	36
3.15	Module "FIRE" de SqueezeNet [8]	37
3.16	Illustration des convolutions groupées (source [9])	39
3.17	Exemple de <i>DeepWise Convolution</i>), avec trois groupes pour trois canaux d'entrée.	39

3.18	Module “Shuffle” de ShuffleNet [10]	40
3.19	Architecture Fast-RCNN [11]	41
3.20	Transformation d'une couche entièrement connectée en convolution pour obtenir une carte d'activation [12]	43
3.21	Exemple d'extraction de caractéristiques depuis une couche cachée d'un réseau de neurones.	45
3.22	Illustration du fine-tuning, avec apprentissage sur première collection et second apprentissage sur la collection cible.	47
3.23	Chronologie sélective de la recherche d'instance (source [13]).	47
3.24	Réseau siamois présenté par [14]	48
3.25	Apprentissage des projection avec un réseau triple (source [15])	50
3.26	Réseau siamois à trois branches	50
3.27	Différentes méthodes de fusion d'information temporelle avec un CNN (source [16]).	52
3.28	Schéma d'un réseau récurrent. A droite la version compression, à gauche la version dépliée sur chaque entrée (source : [17])	53
3.29	Réseaux récurrents pour la detection d'action dans les vidéos à base de CNN et de LSTM (source [18])	54
4.1	Exemple fictif de recherche d'image par le contenu. Le classement représente la similarité entre les images.	56
4.2	Projection d'images dans un plan 2D	57
4.3	Schéma de l'apprentissage du plongement des images, pour une image de références r , avec les images positives p représentant le même objet, et les images n qui contiennent des objets différents.	59
4.4	Exemple des différentes configuration des projections possibles. La projection de l'image de référence r est plus ou moins proche des projection de l'image positive p ou de l'image négative n	61
4.5	Pipeline d'apprentissage. L'étape 1 correspond à l'apprentissage du réseau sur une grande collection d'image, ici ImageNet. L'étape 2 est le fine-tuning sur la collection du musée. La troisième étape consiste à apprendre à l'aide du réseau siamois triple.	63
5.1	Exemple d'images avec les heat-map des activations maximales, obtenues à partir d'un ResNet-152 après un apprentissage fin. Le réseau est appliqué sur toute l'image de manière strié. Les labels obtenus sur les zones d'activation maximales sont également indiqués.	71

5.2	Exemple d'équivalence entre une couche entièrement connectée et une couche de convolution. Le nombre de paramètres est le même, ici $4 * 2$, ils peuvent donc être copié pour obtenir les mêmes résultats. On peut appliquer la convolution avec une entrée plus grande, en une seule passe.	72
5.3	Calcul de la fonction de coût lorsque l'on entraîne le réseau sur différentes régions à différentes échelles. L'entropie croisée E est calculé pour chaque segment d'image avant d'être moyenné (equation. 5.2).	73
5.4	Architecture du réseau proposé basé sur un réseau entièrement connecté pré-entraîné pour détecter les régions d'intérêt.	74
5.5	Pipeline final d'apprentissage, avec l'ajout de la détection des régions d'intérêts.	76
5.6	Exemples de succès et d'échecs de notre approche. La première colonne représente les requêtes, les deuxièmes colonnes contiennent les images les plus proches et la troisième colonne la deuxième image la plus proche dans la collection CLICIDE.	79
6.1	Convolution $3 * 3$ DeepWise suivie d'une convolution $1 * 1$ qui fait office de fusion.	83
6.2	Schéma du module S1.	85
6.3	Schéma du module S2.	87
6.4	Schéma du réseau proposé pour la reconnaissance de geste, avec entre chaque connection la taille de la matrice de représentation des données.	89
6.5	Bloc de convolution utilisant DeepWise Convolution et FrameWise Convolution.	92
6.6	Bloc S1 et S2 FrameWise.	93
6.7	Réseau proposé pour la détection de geste. Les premières couches sont sensibles à la trame (FrameWise), jusqu'à la fusion des N trames par une convolution $1 * 1$	94
6.8	Courbe d'évaluation de la position de la fusion.	95
A.1	Images tirées de la collection Clicide. de gauche à droite : “Portrait de la mère du docteur Bordier” de Hippolyte Flandrin, “Les fruits” de Séraphine de Senlis, “O Combate” de Vicente do Rego Monteiro.	104
A.2	Photographies d’“Animaux Fleurs et Fruits” d’Alexandre-François Desportes tirée de Clicide.	105
A.3	Photographies de la collection de Fourvière. De gauche à droite : une stèle, une statue et une poterie.	106

A.4	Interface d'annotation des vidéos. En haut : l'affichage de la vidéo, en bas chaque segment annoté de vidéo avec l'identifiant d'objet.	107
A.5	Les 5 gestes à reconnaître	109
A.6	Installation du fond-vert (www.videomaker.com)	110
A.7	Le même geste fait par différentes personnes. Peu d'instructions ont été données pour l'apparence désirée du geste	111
A.8	Le geste capturé sur un fond vert, avec un arrière-plan intégré et la segmentation automatique	111
B.1	Répartition des groupes de personnes en fonction de leur habitude de visite.	114
B.2	Gestes Sélectionner et Démarrer identifiés lors des séances participatives	115
B.3	Gestes Sélectionner et Démarrer identifiés lors des séances participatives	116
C.1	Fusion 1	122
C.2	Fusion 2	123
C.3	Fusion 4	124
C.4	Fusion 5	125

1

Introduction

1.1 LE CONTEXTE DE CETTE THÈSE

Les sites touristiques, et notamment les musées, ont su assimiler les évolutions technologiques et proposer de nouvelles méthodes d’interaction pour enrichir les visites et les rendre davantage personnalisées. La visite d’un site touristique peut être accompagnée d’un guide, qu’il s’agisse d’un humain (un guide culturel), ou d’une aide électronique. Ces nouveaux outils, pouvant être mobiles ou fixes comme des bornes multimédia, ne remplacent pas un guide humain, mais permettent de fournir des informations visuelles (visoguide) ou des informations audio (audio-guide), tout en réduisant la fatigue liée à la visite de musée sans aide [19]. Grâce aux avancées techniques, telles que la réduction de la taille des capteurs ou la puissance des processeurs mobiles, les possibilités des outils mobiles s’enrichissent. Et permettent même l’arrivée de solutions à base de réseaux de neurones profonds [20].

Les guides électroniques permettent d’influencer les visites de musées, le visiteur restant plus longtemps et analysant plus en détail les œuvres [21]. Malgré le fait qu’ils puissent, dans certains cas, gêner l’interaction sociale des visiteurs en groupes [21, 22], il est au contraire possible, grâce à un bon design, de promouvoir les échanges [23]. En examinant les impacts des nouveaux usages des visiteurs [24], nous pouvons imaginer de nouvelles



Figure 1.1 : Exemple d'une aide à la visite avec un smartphone. L'utilisateur peut prendre en photo une œuvre pour avoir des informations sur celle-ci.

pratiques, avec notamment des visites plus personnalisées. Que ce soit à travers la réalité augmentée ou l'identification automatique du contexte dans lequel se trouve l'utilisateur, ces systèmes reposent sur des procédés capables de reconnaître l'environnement qui entoure l'utilisateur. Nous utiliserons dans la suite indifféremment utilisateur ou visiteur, car nous nous situons dans l'environnement de la visite de musées.

Cette thèse s'inscrit, avec un financement européen, dans le cadre d'un projet du Fonds Unique Interministériel (FUI) : Guide Multimédia de Tête, Informatif et Connecté (GUIMUTEIC). Ce projet a pour but de proposer une nouvelle génération de guides pour les visites de musée. Le dispositif GUIMUTEIC est équipé de différents capteurs, notamment une caméra. Il est prévu pour être porté par chaque utilisateur et est destiné à enrichir la visite, en la personnalisant et en facilitant l'interaction. Pour cela, le système GUIMUTEIC doit être capable d'identifier les points d'intérêts regardés par l'utilisateur, de le renseigner lorsque celui-ci le désire, et de réagir en fonction de ses actions. Ce travail de thèse explore donc les besoins d'identification d'œuvres, d'actions et de l'environnement de l'utilisateur.

1.2 PROBLÉMATIQUES DE LA RECHERCHE D'INFORMATION MUSÉALE

Les problématiques soulevées portent sur l'aide pouvant être apportée à l'utilisateur et les moyens d'interactions avec le dispositif mobile. Équipé du dispositif GUIMUTEIC, intégrant une caméra embarquée, le but est de fournir à l'utilisateur une information sur son environnement, notamment les œuvres se trouvant face à lui. Pour cela, il est nécessaire d'utiliser des méthodes d'identification d'instances précises. Nous référons ici par “*Ins-*

tance”, l’instance d’un objet, par exemple la Joconde, par opposition à une classe d’objet, dans ce cas un tableau. Nous avons également besoin de déterminer si, et à quel moment, l’utilisateur désire avoir accès à de l’information à propos d’une œuvre. En prenant comme exemple le musée de Bibracte¹, où certaines actions se déclenchent quand le visiteur entre dans une pièce, on comprend l’intérêt de réagir en fonction des actions de l’utilisateur. La localisation de l’utilisateur seule n’est pas toujours suffisante lorsque l’on s’intéresse à identifier son environnement, en ne prenant pas en compte l’orientation ou ce à quoi l’utilisateur s’intéresse [25]. C’est pourquoi nous allons nous intéresser à détecter des points d’intérêts que le visiteur regarde, en plus d’identifier ses actions grâce à la détection de geste par le dispositif GUIMUTEIC.

En dehors des problématiques scientifiques énoncées ci-dessus, ce travail s’inscrit dans le cadre d’un projet “industriel”. Ceci définit un certain nombre de contraintes qui vont influencer les choix effectués par la suite. Tout d’abord, le système GUIMUTEIC est à destination de sites touristiques pouvant être de natures différentes, que ce soit en plein air ou en intérieur. De même, le type des œuvres présentes dans un musée est très variable, de pierres taillées dans un musée d’archéologie, à des peintures ou des œuvres abstraites dans un musée d’art moderne, en passant par des voitures ou des pièces de monnaie. L’objectif est de fournir une solution générale, pouvant s’adapter à chacun.

Une autre contrainte, directement liée au caractère industriel du projet, est l’effort de mise en place dans chacun des musées, qui doit être minimisé. L’acquisition des données, la quantité de données recueillies ou l’annotation de ces données, sont des opérations coûteuses qui doivent être limitées. L’objectif étant de proposer une solution applicable aisément à chaque musée, le travail demandé pour la mise en place doit être aussi réduit que possible. L’étude de ces contraintes et de leurs impacts sur les choix pouvant être fait sera présentée dans le chapitre 2 intitulé “Les enjeux de l’interaction dans les visites touristiques”.

1.3 CONTRIBUTIONS

Les objectifs de cette thèse sont multiples : nous nous intéresserons à la reconnaissance d’instances et à l’exploration de différentes méthodes d’interaction avec l’utilisateur, tout en respectant les contraintes imposées par le projet GUIMUTEIC.

Ce travail de thèse apporte les contributions suivantes :

Étude de l’accès à l’information pendant une visite touristique La première contribution de ce travail de thèse concerne l’étude des visites de musées et les besoins

1. <http://www.bibracte.fr/>

en terme d'accès à l'information au cours des visites touristiques. Nous proposons un système mobile d'interaction entre l'utilisateur et le dispositif, qui permet de répondre aux attentes des musées et du projet GUIMUTEIC.

Reconnaissance d'instances Nous proposons également une approche pour l'identification d'instances, dans le but de reconnaître les scènes vues par l'utilisateur. Pour cela, nous présentons un nouveau système de recherche d'image à base de réseaux de neurones siamois, pour l'apprentissage de similarité entre les images,.

Apprentissage des régions d'intérêt Dans le but d'améliorer la reconnaissance d'instances, nous exposons une nouvelle méthode d'apprentissage automatique des régions d'intérêt dans les images. Cette méthode se base sur un apprentissage non supervisé des zones de l'image, et ne nécessite pas d'annotation de régions sur les images. Nous étudions l'apport de cette détection sur l'apprentissage de similarité, et comment apprendre les deux conjointement.

Détection de geste Pour proposer une interaction avec l'utilisateur, nous cherchons à détecter ses gestes et actions grâce au dispositif GUIMUTEIC, c'est-à-dire avec une vue égocentrique (à la première personne). Nous développons pour cela un nouveau système de reconnaissance de gestes dans les vidéos à base de réseaux de neurones à convolutions, sans récursion.

Création de corpus d'évaluations Pour vérifier la faisabilité du système développé dans des conditions réelles, nous avons recueilli plusieurs corpus dans les musées partenaires du projet. Tous les algorithmes développés dans cette thèse sont évalués sur ces corpus. La collecte et l'organisation de ces corpus sont présentées en annexe A.1 intitulée "Corpus d'images pour la recherche d'instance dans un musée d'art".

1.3.1 PUBLICATIONS

Le travail développé dans cette thèse à fait l'objet des publications suivantes :

- Maxime Portaz, Matthias Kohl, Jean-Pierre Chevallet, Georges Quénnot, and Philippe Mulhem. Object instance identification with fully convolutional networks. *Multimedia Tools and Applications*, pages 1–18, 2018
- Maxime Portaz, Matthias Kohl, Georges Quénnot, and Jean-Pierre Chevallet. Fully convolutional network and region proposal for instance identification with egocentric vision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2383–2391. IEEE, 2017
- Maxime Portaz, Johann Poignant, Mateusz Budnik, Philippe Mulhem, Jean-Pierre Chevallet, and Lorraine Goeuriot. Construction et évaluation d'un corpus pour la

- recherche d’instances d’images muséales. In *CORIA*, pages 17–34, 2017
- Maxime Portaz, Philippe Mulhem, and Jean-Pierre Chevallet. Étude préliminaire à la recherche de photographies muséales en mobilité. In *CORIA 2016 COnférence en Recherche d’Information et Applications 2016*, pages 335–344, 2016
 - Maxime Portaz, Mateusz Budnik, Philippe Mulhem, and Johann Poignant. Mrim-lig at imageclef 2016 scalable concept image annotation task. In *CLEF (Working Notes)*, pages 363–370, 2016

Le code développé ainsi que les collections créées dans cette thèse sont librement disponibles. Pour le code utilisé pour les expérimentations, et également les schémas et les tableaux présentés, sont disponibles sur GitHub à l’adresse <https://github.com/maxgreat>. Les collections d’images et de vidéos sont disponibles en accès libre dans le cadre de la recherche à l’adresse <http://lig-mrim.imag.fr>.

1.4 PLAN DU MANUSCRIT

Afin de décrire le travail mené dans le cadre de cette thèse nous suivons le plan suivant :

Chapitre 2 : Les enjeux de l’interaction dans les visites touristiques Nous réalisons tout d’abord une étude approfondie du problème d’accès à l’information dans le cadre de visites touristiques. Nous analysons plus en détail les contraintes spécifiques des musées. Nous nous intéressons particulièrement aux différents moyens d’accès à l’information possibles dans ce contexte, et notamment aux types d’interactions avec l’utilisateur envisageables et désirables pour les visiteurs.

Chapitre 3 : État de l’art Nous introduisons différentes méthodes de recherche d’informations multimédia et d’analyse d’images existantes dans la littérature. Nous présenterons l’état de l’art de la recherche d’images, particulièrement les techniques de réseaux de neurones profonds (Deep Learning). Un état de l’art de la détection d’événements dans les vidéos est aussi présenté.

Chapitre 4 : Identification d’œuvres Dans ce chapitre, nous présentons l’analyse et la modélisation du problème d’identification d’œuvres dans les images et vidéos. Nous explorons différentes méthodes basées sur ce qui a été présenté dans le chapitre 3, et nous les adaptons à nos problématiques. Nous évaluons ces méthodes sur les corpus d’images construits.

Chapitre 5 : Detection de régions d’intérêts Nous fournissons une nouvelle méthode pour la détection de régions d’intérêt dans les images. Grâce à un apprentissage de la localisation des objets dans les images, sans annotation des régions sur le corpus

d'apprentissage, nous obtenons des résultats supérieur à l'état de l'art sur nos collections.

Chapitre 6 : Détection de gestes Ce chapitre est consacré à l'étude de la détection des actions de l'utilisateur dans les vidéos. Suite à l'étude réalisée dans le chapitre 2, nous nous intéressons aux gestes nécessaires à l'interaction et aux méthodes de détection possibles. Nous étudions la détection des gestes avec une vue à la première personne. Nous utilisons des réseaux de neurones non récursifs, avec pour objectif d'avoir un réseau de neurones compact et rapide, utilisable sur mobile. Nous évaluons notre approches grâce à un corpus de vidéos créé spécialement.

Chapitre 7 : Conclusion Finalement, le dernier chapitre présente la démarche générale. Nous concluons sur les contributions de ce travail de thèse, notamment sur les méthodes d'identifications d'instance et de détection de gestes à l'aide de réseaux de neurones profonds et sur la création du système GUIMUTEIC, ainsi que sur la suite des recherches possibles sur les problématiques présentées.

2

Les enjeux de l'interaction dans les visites touristiques

L'objectif de ce travail de thèse est de répondre aux problématiques d'accès à l'information durant les visites touristiques. Ce chapitre va étudier en détail les besoins et les attentes des musées et des visiteurs dans le cadre des nouveaux types d'interaction existants à l'heure actuelle. En s'intéressant notamment à une interaction à base de gestes, développée grâce à une étude utilisateur réalisée avec des séances participatives.

Les raisons de créer de nouveaux moyens d'interaction dans les musées sont multiples (section 2.1), ce qui implique de réfléchir à de nouvelles méthodes d'accès à l'information. Le projet GUIMUTEIC (section 2.2) doit répondre à ces besoins, tout en respectant ses contraintes de déploiement.

2.1 LES NOUVEAUX MOYENS D'INTERACTION ET D'ACCÈS À L'INFORMATION MUSÉALE

Les professionnels des musées, ainsi que les visiteurs, ont vu leur compréhension de ce qu'est un musée changer, avec l'introduction de nouvelles technologies au sein des musées [31]. Le problème des nouvelles interactions n'est pas que technologique, il nécessite

une étude interdisciplinaire des interactions sociales et technologiques dans les musées, comme le souligne le “museum informatics” défini par Marty [32].

L'essor des sciences de l'information et des technologies a créé de nouveaux moyens d'accès et de traitement de l'information dans les musées, et une nouvelle manière de penser les musées est apparue [33]. Les musées ont traditionnellement servi de recueil d'objets, dans le but de les préserver, de les étudier ou d'éduquer. Cependant, ils ne sont pas là uniquement pour stocker des objets. Même si la partie fondamentale d'un musée reste sa collection d'œuvres et d'objets, la grande quantité d'information disponible à propos de chacun de ces objets est tout aussi importante [34]. Les visiteurs s'attendent aujourd'hui à avoir un accès instantané à toutes les connaissances du musée à propos de chaque œuvre. Pour rendre cela possible, il a été établi depuis longtemps que les musées se devaient de proposer de nouveaux moyens de gérer l'information [35]. Chaque visiteur est unique, avec des attentes différentes, et ce qu'il retiendra d'une visite va dépendre des objets vus, du personnel du musée, du contexte socioculturel et de certains aspects propres à la personne [36]. Il est important d'avoir une personnalisation de la visite pour que chacun puisse avoir la meilleure expérience, adaptée à ces besoins. Par exemple, le projet PIL [37] propose une réflexion sur comment adapter le musée au visiteur, grâce aux outils technologiques, tout en faisant en sorte que celui-ci soit concentré sur les expositions et non pas sur les appareils de visites. Il faut notamment faire attention à ce que les outils technologiques ne bloquent pas la communication avec les autres visiteurs. Ils ne doivent pas être invasifs pendant la visite, pour que l'utilisateur puisse se concentrer sur les œuvres.

Il est commun aujourd'hui d'avoir un audio-guide (ou un visio-guide) dans les musées afin d'avoir une audio(visio)-description d'une œuvre d'art, son impact sur la visite n'est donc pas à négliger. Les guides permettent de garder les utilisateurs plus longtemps à l'intérieur du musée, en passant plus de temps à étudier les œuvres, tout en s'intéressant davantage d'œuvres [21]. Ils permettent également d'améliorer l'interaction avec l'utilisateur [38] ainsi que de renforcer le côté éducatif de la visite [39]. Cependant, les guides électroniques ont tendance à fermer les personnes à l'interaction sociale [40]. Dans le cas de visites en groupes, les appareils mobiles forcent l'utilisateur à regarder l'écran et à ne pas faire attention aux personnes avec lesquelles il est venu [22, 41]. Il devient également fréquent d'avoir accès à l'information directement sur son smartphone. Ceux-ci ont le problème de couper le visiteur de ce qui l'entoure, car il va regarder son téléphone plutôt que ce qui l'entoure. Comme il est devenu clair qu'il était difficile de détourner une personne de son smartphone, celui-ci peut être utilisé par le musée pour guider le visiteur ou fournir un plus à la visite [23, 42, 43]. Pour ne pas détourner l'attention des utilisateurs envers les œuvres, ni couper le visiteur de son groupe, le projet GUIMUTEIC n'utilise pas d'écran,

mais uniquement un casque audio ouvert.

Plusieurs approches modernes proposent de nouvelles interfaces technologiques, pour l'interaction avec les œuvres culturelles. Par exemple, le "SmartMuseum" [44] propose à l'aide de capteurs RFID (Radio Frequency Identification) de donner des informations au visiteur. Ce dernier peut construire sa propre visite sur le site web en amont de sa venue, et reconnaître sur site les œuvres sélectionnées. L'utilisation d'étiquettes RFID limite, ou complexifie, la modification de l'organisation du musée. Le maintien et l'organisation de la base d'étiquette devient une contrainte et demande un certain engagement de la part du musée. Ce système requiert l'utilisation d'un appareil mobile et de plus requiert un certain degré d'apprentissage dans son utilisation.

Une solution pour aider le visiteur est d'utiliser un appareil mobile qui sert de guide audio-visuel, comme le propose le "Museum Wearable" [45]. L'appareil utilise la connaissance de l'environnement de l'utilisateur pour lui fournir l'information sur ce qui l'intéresse à un instant donné. Ce prototype utilise uniquement la position géographique de l'utilisateur pour déterminer son environnement, or la position seule n'est pas nécessairement suffisante pour déterminer le contexte autour de l'utilisateur [25]. Une caméra par exemple permet d'obtenir plus d'information, et si celle-ci est positionnée de manière à voir ce que l'utilisateur perçoit, avec une vue à la première personne. On est alors capable d'avoir une information précise sur l'environnement direct.

Pour déterminer ce qui intéresse le visiteur, des méthodes d'interaction à base de gestes ont été développées. L'approche à base de systèmes portatifs avec vision à la première personne [1] permet de voir ce que l'utilisateur perçoit, de réagir à ces gestes et d'avoir des informations précises sur son environnement. Comme montré sur la figure 2.1a, un utilisateur, équipé de lunettes avec caméra embarquée, peut interagir avec celles-ci grâce à une série de gestes prédéfinis (figure 2.1b). Ici les gestes sélectionnés sont Approuver ("OK"), Désapprouver ("Dislike"), Défiler ("Slide"), Valider ("Like"), Pointer et Prendre une photo ("Take a picture"). Ces gestes n'ont cependant pas été développés avec une étude utilisateur. Ce type d'approches permet de dépasser les limitations des appareil mobiles qui requièrent une manipulation et qui détourne l'attention de l'utilisateur des œuvres. Ceci permet également de proposer de nouvelles expériences d'interaction, et d'encourager le visiteur à interagir avec les œuvres, comme ils le feraient devant un guide humain. Si de tels gestes sont utilisés, il faut cependant étudier l'utilité des gestes reconnus, et lesquels sont les plus à même de correspondre aux attentes des utilisateurs. Il faut ensuite déterminer à quelle moment de la visite réaliser la reconnaissance d'environnement de l'utilisateur, et notamment reconnaître les œuvres en face de lui, pour lui fournir une réponse à son inter-

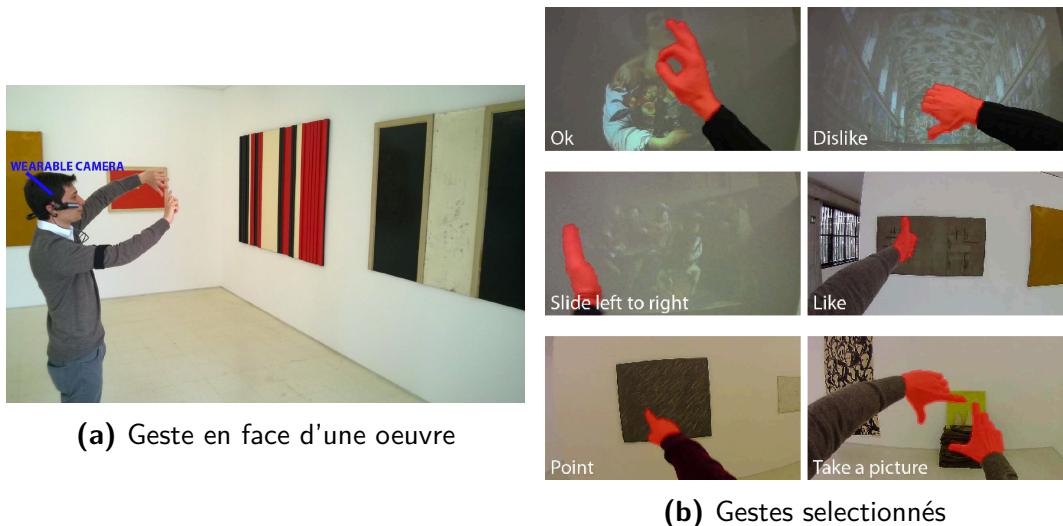


Figure 2.1 : Exemple d'un système basé sur l'interaction à base de geste dans un musée [1]

action.

2.2 LE SYSTÈME GUIMUTEIC

Le système GUIMUTEIC est un système mobile, doté d'une caméra, aidant et améliorant la visite de musées. Pour cela, il doit être capable d'identifier les actions de l'utilisateur, et de réagir à ses besoins. Il doit également pouvoir reconnaître l'environnement de ce dernier, pour le guider et lui apporter les informations correspondantes à ce qu'il regarde.

Avec une série d'études participatives, nous avons identifié les types d'interaction adaptées au projet (paragraphe 2.2.1). Ainsi, tout en respectant les contraintes liées au projet (paragraphe 2.2.2), nous avons pu mettre au point un système d'aide au visiteur qui répond à ses besoins (paragraphe 2.2.3).

2.2.1 UNE INTERACTION À BASE DE GESTES

Des séances participatives de conception d'un appareil d'aide à la visite de musée ont été organisées. Regroupant des participants de différents âges et antécédents, elles ont permis, grâce à des exercices ludiques, de définir les usages futurs du dispositif, et de mettre en avant ce que les utilisateurs finaux attendent du produit. Ces séances ont fait ressortir plusieurs méthodes d'interaction susceptibles d'être intéressantes pour les visiteurs. Les détails de cette étude sont reportés dans l'annexe B "Etudes de la pertinence de l'interaction à base de geste".

À l'issue de ces séances, l'interaction privilégiée est celle à base de gestes. Cinq gestes ont été identifiés comme utiles à l'interaction, et sont présentés dans la figure 2.2. Il s'agit de **pointer**, signaler “OK”, **valider** (ou “liker”), **arrêter** et **stop** (ou pause).

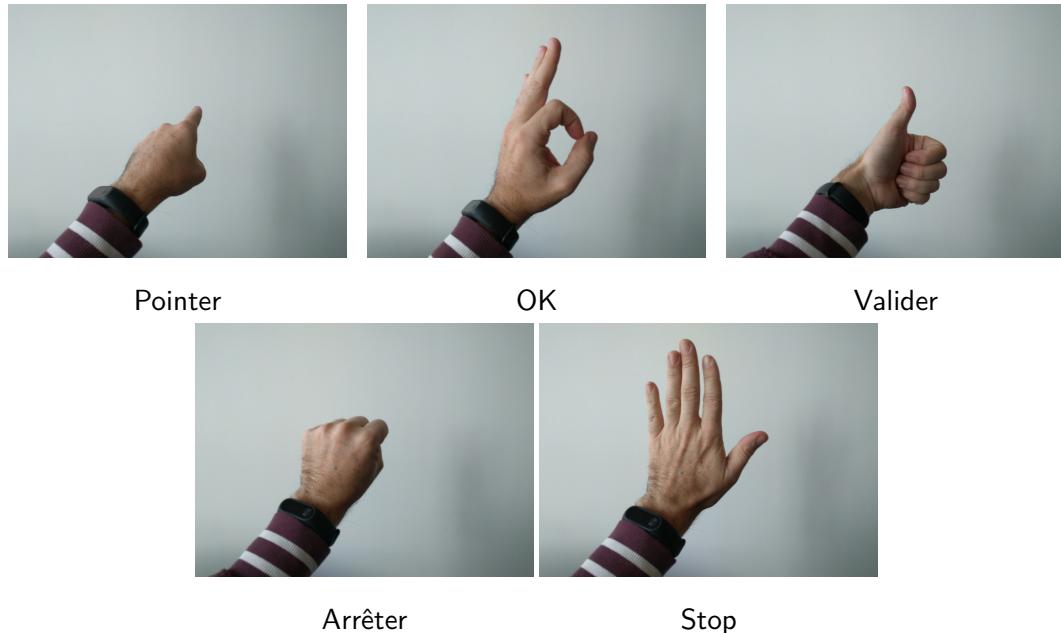


Figure 2.2 : Les 5 gestes mis en avant par les séances de conceptions participative.

Pour créer un système de reconnaissance de ces gestes, un ensemble de vidéos de gestes correspondant aux attentes du projet ont été réalisés. Ce corpus est composé de vidéos en vue à la première personne, prise dans différentes conditions, avec des environnements différents et notamment sur fond vert. La description complète du corpus est fournie en annexe B. Les méthodes proposées dans le chapitre 6 seront testées sur ce corpus.

2.2.2 LES CONTRAINTES DES MUSÉES ET DU PROJET GUIMUTEIC

Le projet GUIMUTEIC est le fruit d'une collaboration industrielle, et doit donc répondre à un certain nombre de contraintes qui en découlent. Le premier type de contraintes est lié à la distributivité du produit. Le système doit être capable d'être applicable à n'importe quel type de musée. Dans le cadre du projet, deux types de musées sont étudiés : un musée d'archéologie, le musée Gallo-Romain de Lyon Fourvière, et le musée d'art de Grenoble. Les types d'objets présents dans ces musées étant variés, leur étude nous permet de produire un système généralisable.

Un autre facteur important de la faisabilité de ce type de projet, est le coût, en infrastructure et en heures, de l'installation du système dans un musée. Compte tenu de l'aspect

portatif de dispositif GUIMUTEIC, il est important qu'aucune connexion permanente à un serveur ne soit nécessaire. En effet, la connection à un serveur distant sous-entend, en plus de la présence et de la maintenance de serveurs, une connectivité dans l'ensemble du musée.

Pour être utilisable par le plus de personnes possible, un guide doit pouvoir être utilisé sans apprentissage de la part de la personne qui l'utilise, pour ne pas décourager celle-ci, et permettre à n'importe qui de l'utiliser de manière simple, sans formation. De plus, le système doit fonctionner sans apprentissage de la part du système pour chaque utilisateur. Il doit être capable de s'adapter à toutes les morphologies, de taille notamment, et de couleur de peau.

De la même manière, pour la mise en place de la reconnaissance d'environnements et d'œuvre autour de l'utilisateur, une base de données des lieux, œuvres et autres points d'intérêt du musée, doit d'être constituée. Ceci pouvant être un frein considérable à l'installation d'un nouveau système, du fait du coût en heures de prise de vues et en maintenance, nous avons de forte contraintes sur la création de cette base. Nous nous limitons à quelques photos par œuvre et point d'intérêt du musée. La capture de vidéos de visite est un processus très coûteux : en plus de l'enregistrement, qui doit faire intervenir un certain nombre de personnes pour avoir une diversité suffisante, il faut prendre en compte le coût de l'annotation de chacune de ces vidéos. Identifier quel objet est visible à chaque instant, si possible même annoter la position de l'objet dans l'image, est long et nécessite un certain investissement, qui ne correspond pas au projet dans lequel nous nous situons. Les propositions dans la suite de cette thèse doivent donc répondre à ces contraintes :

- Quelques images par œuvre : la reconnaissance d'instance dans les images doit pouvoir se faire avec seulement quelques exemples de ladite instance comme référence.
- Portabilité de l'appareil : la quantité de calcul et l'utilisation de la mémoire pour le reconnaissance d'actions et d'instances doivent être aussi limités que possible, en ayant comme objectif de tout réaliser sur un processeur mobile.
- Pas de vidéos de visite : la réalisation de vidéos de visite complètes, annotées, demande trop d'investissement. Nous partons donc du principe que nous ne disposons pas de ce type de vidéos.

2.2.3 LE SYSTÈME D'INTERACTION GUIMUTEIC

Notre approche pour la création d'une aide à la visite muséale est présentée dans le schéma 2.3. L'utilisateur, libéré de tout appareil dans ses mains, peut interagir avec le système GUIMUTEIC à travers des gestes prédéfinis. On voit sur la figure le visiteur sur la gauche interagir avec le système GUIMUTEIC au moyen de la caméra. A l'aide de la re-

connaissance de gestes et de reconnaissance d'instance, le système peut fournir un retour d'information adapté. Nous développons donc une approche de reconnaissance de gestes avec une caméra en vue à la première personne (chapitre 6, Détection de gestes). Dans le but de donner des informations au visiteur, le système doit être capable de détecter son environnement. Nous nous basons pour cela sur la reconnaissance de point d'intérêts dans le musée (chapitre 4 Identification d'œuvres). Ces points d'intérêts sont généralement des œuvres. Le retour d'information à l'utilisateur se fait grâce à un casque audio. Dans le chapitre suivant, nous présentons l'état de l'art de la reconnaissance d'instances dans les images, ainsi que celui de la reconnaissance de gestes et d'actions dans les vidéos.

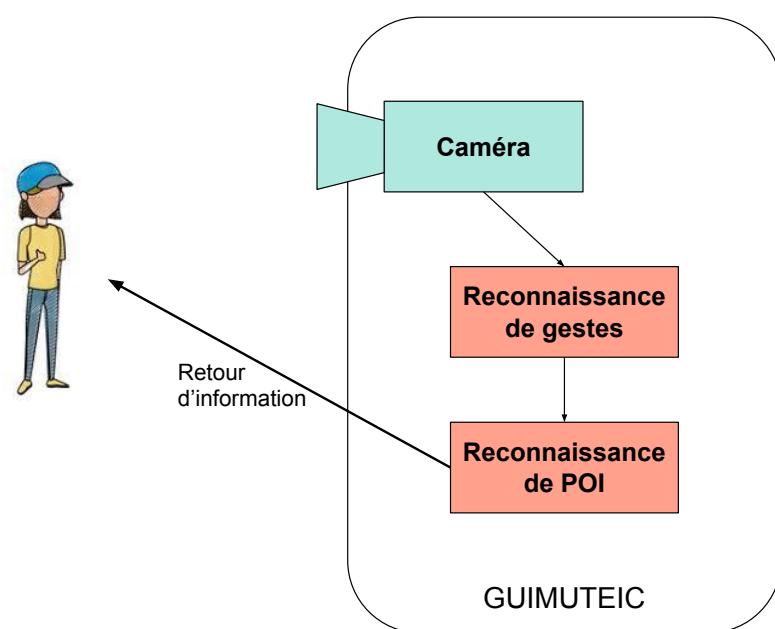


Figure 2.3 : Schéma d'organisation du système d'information GUIMUTEIC.

3

État de l'art

Les réseaux de neurones profonds, de part leur nature, permettent d'apprendre des fonctions d'abstraction de haut niveau pour de nombreux problèmes d'intelligence artificielle [46]. Depuis leur introduction au début des années 90 [47], les réseaux de neurones profonds ont obtenus des résultats impressionnantes dans beaucoup de domaines, comme la détection d'objets [48–50], la reconnaissance de visages [15], de parole [51, 52] ou d'émotions [53]. Les avancées des réseaux profonds ont continué, mais ces dernières années ont vu leur application à un grand nombre de domaines, ceci peut être expliqué par la convergence de plusieurs facteurs. La démocratisation des GPU (Graphics Processing Units) et leur application aux réseaux de neurones qui ont divisés le temps de calcul d'un facteur de 70 à 100 [54] : un apprentissage, qui prenait des semaines à se réaliser, même distribué sur un ensemble de CPU (Central Processing Units), peut être effectué en quelques heures. L'arrivée des ReLU (Rectified Linear Units) a réduit le problème de la disparition du gradient (Vanishing Gradient Problem) [55]. Enfin, la création de grandes bases de données comme ImageNet [56], ont permis d'apprendre des réseaux plus profonds et plus performants.

Ce chapitre présente l'état de l'art de la recherche d'information multimédia. Aujourd'hui basée essentiellement sur les réseaux de neurones profonds, la majorité des éléments développés dans cette thèse reposent sur ceux-ci. Nous présentons donc dans un premier

temps les réseaux de neurones profonds convolutifs (section 3.1), qui seront largement utilisés dans la suite. Nous nous intéressons notamment à la recherche et l'identification d'instances à l'aide de réseaux de neurones (section 3.5). Enfin, nous étudions la détection et l'annotation de séquences dans les vidéos (section 3.6), ce qui nous sera utile pour la détection de geste (dans le chapitre 6), avant de conclure sur ce qui motive les recherches présentées dans cette thèse.

3.1 RÉSEAUX DE NEURONES PROFONDS

Pour comprendre la suite de cette thèse, nous introduisons certaines notions sur les réseaux de neurones, à commencer par la définition d'un neurone. Bien qu'inspiré par la biologie, un neurone en informatique est une version largement simplifiée de son modèle d'origine.

3.1.1 NEURONES

On peut voir un neurone comme une fonction non linéaire N qui prend en entrée un certain nombre d'éléments, et renvoie une valeur. Soit la fonction linéaire f :

$$f(x) = \sum_i^n w_i x_i + b \quad (3.1)$$

f est une somme pondérée des entrées $x = x_0, , x_n$ avec les poids $W = w_0, , w_n$, à laquelle on ajoute le biais b . Si chaque neurone n'était qu'une fonction linéaire, alors n'importe quel réseau constitué d'un ensemble neurones connectés serait en fait équivalent à un seul neurone, ou une seule couche de neurones, car il ne réaliseraient au final qu'une combinaison linéaire des entrées. Pour qu'un réseau de neurones réalise des fonctions plus complexes, il est nécessaire d'ajouter de la non-linéarité sur les neurones. Un neurone réalise la fonction f et ajoute une fonction non-linéaire σ . Cette fonction σ est appelée fonction d'activation, en rapport avec le potentiel d'action dans les neurones en biologie, et va définir la sortie du neurone.

$$N(x, W) = (\sigma \circ f)(x, W) = \sigma(\sum_i w_i x_i + b) \quad (3.2)$$

L'équation 3.2 est illustrée par la figure 3.1, qui montre un neurone connecté aux trois entrées à gauche. Sur chaque connexion est affiché le poids associé, qui l'on pourrait schématiser comme étant l'importance donnée à cette connexion par ce neurone. Le somme de ces entrées est passé à travers la fonction d'activation, ici la fonction sigmoïde, pour calculer

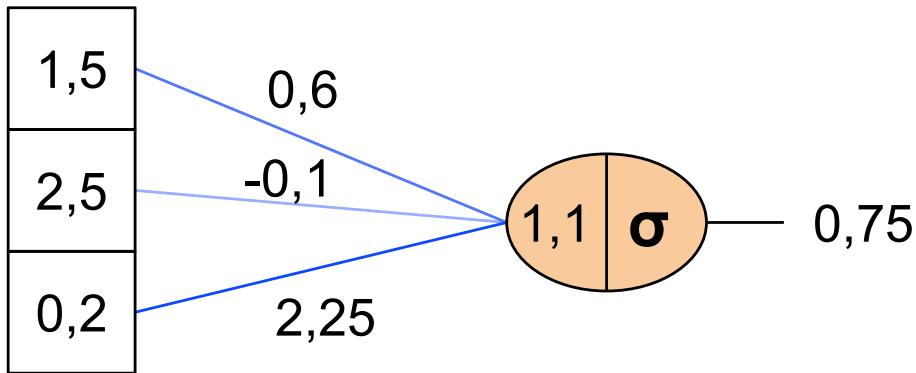


Figure 3.1 : Schéma d'un neurone avec trois connexions entrantes.

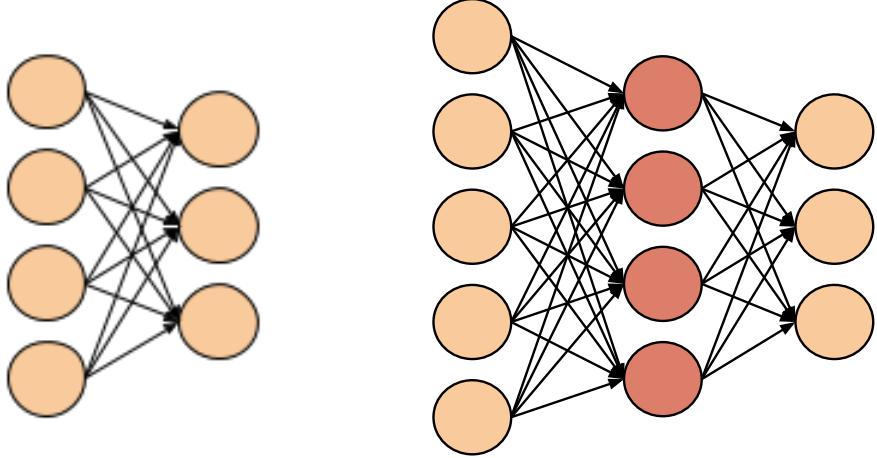
la sortie du neurone.

3.1.2 PERCEPTRON MULTI-COUCHE

Avec ce modèle simple de neurones, on peut définir un ensemble de neurones connectés les uns aux autres. Un réseau de neurones profond entièrement connecté à plusieurs couches, appelé Perceptron multicouche, est un ensemble de neurones connectés, organisés en couches successives. Sur la figure 3.2a, nous voyons une couche de neurone entièrement connectée (en anglais Fully Connected Layer), avec un vecteur en entrée de taille 4 et une sortie de taille 3. Une telle configuration avec i neurones, produisant la sortie y de taille i grâce à l'entrée x de taille n , peut être calculé de la manière suivante :

$$\forall y_i \in y, y_i = \sigma \left(\sum_{j=0}^n (x_j * w_j + b_i) \right) \quad (3.3)$$

Une couche ainsi définie est une classifieur linéaire, qui réalise sa prédiction grâce à une combinaisons des entrées. Pour réaliser des fonctions plus complexes, des modèles avec une couche profonde sont nécessaires. La figure 3.2b représente un modèle avec deux couches entièrement connectées, avec ce que l'on nomme une couche cachée, en rouge sur le schéma. Ces réseaux n'ont pas de connexions depuis la sortie vers l'entrée, ils sont nommés réseau à propagation avant (en anglais “feed-forward networks”). À l'inverse les modèles avec des connexions allant de la sortie vers l'entrée sont appelés réseaux récurrents



(a) Réseau à 1 couche

(b) Réseau à 2 couches

Figure 3.2 : Réseaux de neurones à 1 et 2 couches.

(cf section 3.6). Chaque couche peut avoir une dimension arbitraire, et on remarque que le nombre de paramètres du réseau augmente rapidement. Sur l'exemple de la figure 3.2b, nous avons deux couches, la première ayant $5 * 4 + 4 = 24$ paramètres et la deuxième $4 * 3 + 4 = 16$ en comptant les poids et les biais, ce qui fait 40 paramètres à apprendre.

Le théorème d'approximation universelle [57] stipule qu'un réseau de neurones avec une couche cachée peut approximer toute fonction continue sur un sous-ensemble compact de \mathbb{R}^n , à condition d'avoir une fonction d'activation continue, non-constante, bornée et croissante. Ce qui fait d'un perceptron multi-couche un approximateur universel, du moins en théorie, rien ne garanti qu'il soit possible de trouver une telle fonction par apprentissage.

3.1.3 APPRENTISSAGE DES PARAMÈTRES DU RÉSEAU

Pour que notre réseau profond réalise la fonction que l'on désire, il est nécessaire d'apprendre chacun de ses paramètres. L'apprentissage se fait par mise à jour des poids W , par descente de gradient :

$$W = W - \alpha \frac{dL(\hat{y}, y)}{dW} \quad (3.4)$$

où α est la *learning-rate*, qui définit de combien vont être ajusté les poids à chaque mise à jour. $L(\hat{y}, y)$ est la fonction objectif, ou fonction de coût, qui évalue si la sortie y de notre réseau est correcte par rapport à la distribution attendue \hat{y} . $\frac{dL(\hat{y}, y)}{dW}$ est la dérivée partielle de

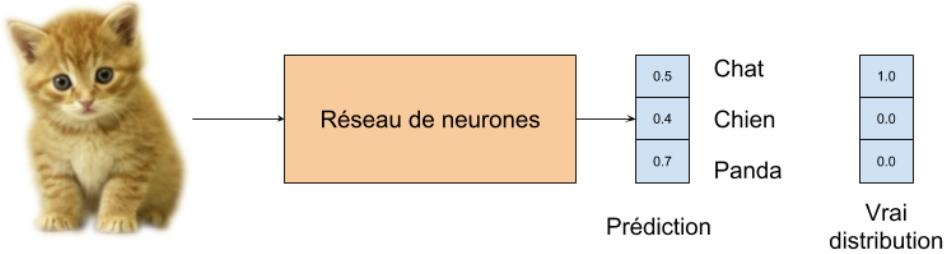


Figure 3.3 : Exemple d'apprentissage d'un réseau de neurone.

la fonction L , par rapport à chaque poids W , qui nous indique dans quel sens mettre à jour les poids (augmenter ou diminuer).

Prenons comme exemple la sortie d'un réseau de classification, présenté sur la figure 3.3. Étant donnée une image, le réseau de neurones va produire une sortie $q = (0.5, 0.4, 0.7)$. La distribution attendue $p = (1.0, 0.0, 0.0)$, correspond aux labels corrects de l'image. On peut alors calculer l'entropie croisée, notée CE , grâce à l'équation 3.12, ce qui donne :

$$CE(p, q) = -1 * \log(0.5) = 0,693 \quad (3.5)$$

Plus le réseau produit une distribution avec une grande confiance dans la bonne catégorie, plus l'entropie croisée sera proche de 0, et inversement. Minimiser cette fonction par descente de gradient sur l'ensemble du réseau va permettre d'obtenir la bonne distribution en sortie du réseau, ce qui en fait une fonction objectif de choix. On voit cependant que l'entropie croisée ainsi définie ne regarde que le score sur la bonne catégorie pour évaluer si la prédiction est correcte ou non. Des approches récentes proposent ce qu'ils nomment “label smoothing” pour prendre un compte toutes les catégories, comme méthode de régularisation. La régularisation dans les réseaux de neurones est très importante et permet de générer des modèles plus généraux, et d'éviter le sur-apprentissage (cf section 3.1.5).

On parle de convergence si l'évolution de l'ensemble des paramètres est capable de trouver (de converger vers) des valeurs permettant de produire une réponse correcte en sortie du réseau. Pour mettre à jour ces paramètres, on utilise la descente de gradient par lots. Un lot (ou *batch*) est un sous ensemble de l'ensemble des exemples x . C'est un moyen d'organiser les données d'apprentissages en paquets. Si l'on parcourt l'ensemble de ces lots de

taille S , on parle alors d'une *epoch*.

$$W = W - \alpha \frac{1}{S} \sum_{j=1}^S \frac{dL(\hat{y}_j, y_j)}{dW} \quad (3.6)$$

L'utilisation de *batchs* permet d'accélérer l'apprentissage en profitant de la parallélisation, et peut réduire le bruit amené par l'utilisation de chaque exemple indépendamment. On peut contrôler facilement la quantité de mémoire nécessaire pour l'apprentissage en jouant sur S , et le calcul du gradient peut être fait sur le *batch* en entier, ce qui accélère grandement les calculs. La stratégie classique consiste à mélanger aléatoirement les exemples pour chaque *epoch*, et des les organiser en *batch* de taille 16 à 64 [58]. Cependant, il a été démontré que l'utilisation de *batchs* trop grands détériore les résultats et empêche d'avoir une bonne généralisation [59].

Cross-entropy : on utilise principalement l'entropie-croisée *CE* comme fonction objectif, pour les raisons suivantes :

Nous nous intéressons tout d'abord à l'entropie. Prenons la loi de distribution de probabilité p et un ensemble de n exemples (x_1, x_2, \dots, x_n) . L'entropie de Shannon [60] nous indique le nombre de bits nécessaires pour la représentation de p , et est définie par :

$$E(p) = - \sum_{i=0}^n p(x_i) \log p(x_i) \quad (3.7)$$

La dernière couche d'un réseau de neurones peut être un *softmax*, c'est-à-dire avoir des valeurs comprises entre 0 et 1, avec une somme à 1. Dans ce cas, la sortie du réseau peut être vue comme une distribution q qui doit estimer p , la vraie distribution. Nous pouvons calculer la divergence de Kullback-Leibler qui nous donne la différence moyenne du nombre de bits nécessaires pour représenter q par rapport à p . La divergence de Kullback-Leibler entre p et q est définie par :

$$D_{KL}(p, q) = \sum_{i=0}^n p(x_i) \log \frac{p(x_i)}{q(x_i)} \quad (3.8)$$

On veut minimiser la différence entre q et p , ce qui revient à minimiser la somme de l'entropie de p et de la divergence de Kullback-Leibler de q par rapport à p .

$$E(p) + D_{KL}(p, q) = - \sum_{i=1}^n p(x_i) \log p(x_i) + \sum_{i=0}^n p(x_i) \log \frac{p(x_i)}{q(x_i)} \quad (3.9)$$

$$= \sum_{i=1}^n p(x_i) \left(-\log p(x_i) + \log \frac{p(x_i)}{q(x_i)} \right) \quad (3.10)$$

$$= \sum_{i=1}^n p(x_i) (-\log p(x_i) + \log p(x_i) - \log q(x_i)) \quad (3.11)$$

$$= - \sum_{i=1}^n p(x_i) (\log q(x_i)) = CE(p, q) \quad (3.12)$$

Chaque opération réalisée par le réseau étant dérivable, on va mettre à jour les poids par rétro-propagation du gradient, de la sortie vers l'entrée. Ceci n'empêche pas que l'on risque de finir dans un minimum local, c'est-à-dire que la descente de gradient ne permettra plus l'amélioration des performances, alors qu'il existe un minimum global où le réseau serait meilleur. Cependant, en pratique, il n'est pas constaté l'existence particulière de problème avec les minimum locaux, et la plupart des apprentissages donne des résultats similaires, avec des initialisations aléatoires du réseau au départ. Il a d'abord été conjecturé qu'il n'existe pas de minimums locaux [47], mais il a été prouvé que tous les minimum locaux sont des minimums globaux pour les réseaux de quelques couches, bien que quelques minimum non-optimaux puissent exister dans les réseaux plus profonds [61].

3.1.4 NON-LINÉARITÉ ET FONCTIONS D'ACTIVATION

La fonction d'activation des neurones apporte la non-linéarité des connexions. Les fonctions d'activation non linéaires peuvent être très variées et ont évoluées au fur et à mesure des années, mais on peut en relever trois principales. Historiquement, la fonction sigmoïde fut la première proposée, elle est définie par :

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3.13)$$

Cette fonction projette les nombres réels dans $[0, 1]$, plus particulièrement les grands nombres deviennent 1 et les plus petit nombres deviennent 0 (figure 3.4a). La fonction sigmoïde est moins utilisée aujourd'hui car elle a tendance à réduire le gradient à zéro lors de la rétro-propagation. Un autre désavantage de cette fonction est qu'elle n'est pas centrée sur 0. Ce qui signifie que les couches suivantes dans le réseau recevront des données qui ne

sont pas centrée sur 0, ce qui est problématique pour le calcul du gradient. Si on prend la dérivée partielle de L en fonction de w_i :

$$\frac{dL}{dw_i} = \frac{dL}{df} \frac{df}{dw_i} \quad (3.14)$$

or d'après l'équation 3.1 :

$$\frac{df}{dw_i} = x_i \quad (3.15)$$

et donc

$$\frac{dL}{dw_i} = \frac{dL}{df} x_i \quad (3.16)$$

Or si $x_i > 0$, le gradient $\frac{dL}{dw_i}$ sur les poids w_i sera toujours du même signe (positif ou négatif) que $\frac{dL}{df}$. Avec tous les gradients de même signe, tous les poids vont être mis à jour dans le même sens (augmentés ou diminués). Ceci va créer un effet de “zig-zag” lors de la mise à jour des poids, avec une mise à jour dans un sens, puis dans un autre.

Ce problème n'apparaît pas avec la fonction tangente hyperbolique (figure 3.4b) :

$$\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} = 2\sigma(2x) - 1 \quad (3.17)$$

qui correspond en fait à la fonction sigmoïde centrée sur zéro, avec une sortie entre $[-1, 1]$. Cette fonction est généralement à préférer à la fonction sigmoïd [62].

Cependant, la fonction ReLU pour Rectified Linear Unit [55], est celle qui s'est le plus répandue ces dernières années et qui est largement la plus utilisée. Elle est définie comme un seuil à zéro :

$$ReLU(x) = \max(0, x) \quad (3.18)$$

La figure 3.4c présente graphiquement cette équation. Cette fonction accélère grandement la convergence, ce qui serait dû à sa linéarité lorsque f est positive, ce qui évite la saturation comme la tangente hyperbolique [4]. Un avantage considérable de cette fonction est sa facilité de calcul, qui ne requiert qu'un filtrage de la matrice d'activation avec un seuil à 0. Le fait de mettre un certain nombre de neurones à 0 peut inactiver une partie du réseau qui au final ne sera jamais activé. Pour éviter cette “mort” d'une partie du réseau, le Leaky ReLU a été proposé [6], mais avec des résultats peu stables, il ne s'est pas généralisé. Cette fonction, proche du ReLU, ajoute un paramètre α qui doit être appris, lorsque x est

négatif.

$$f(n) = \begin{cases} ax & \text{if } x < 0 \\ x & \text{if } x > 0 \end{cases} \quad (3.19)$$

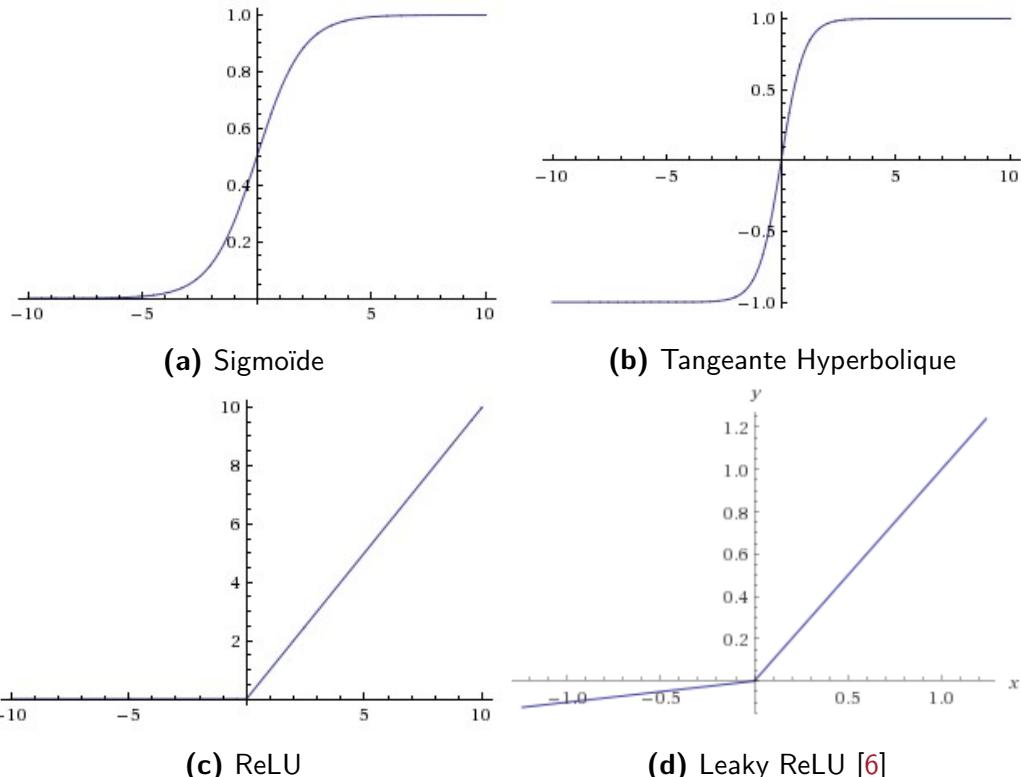


Figure 3.4 : Fonctions d'activation non-linéaires.

3.1.5 RÉGULARISATION

Lorsque l'on réalise un algorithme d'apprentissage automatique, on souhaite que celui-ci n'ait pas uniquement de bonnes performances sur les données d'apprentissages, mais sur toutes données qu'on lui présente. La régularisation est l'ensemble des méthodes qui permettent à l'algorithme de réduire son erreur de généralisation.

Généralisation : capacité d'un réseau de neurones, et de n'importe quel algorithme d'apprentissage automatique, à bien se comporter sur des données qu'il n'a pas vu pendant l'entraînement. Généralement mesuré grâce à un corpus de test séparer du corpus d'entraînement. Un exemple extrême d'un algorithme qui généralise mal, est un réseau capable d'atteindre 100% de reconnaissance sur les données fournies pendant l'apprentissage et 0% sur le corpus de test. On parle de **sur-apprentissage** [2].

Une façon schématique de voir la régularisation est d'imaginer un ajout de bruit dans les réseaux pour les rendre plus robustes et moins spécialisés sur leurs données d'apprentissages. L'idée est donc de déstabiliser le réseau lors de l'apprentissage, pour qu'il ne soit pas limité aux exemples fournis pendant l'apprentissage, dans le but d'éviter un sur-apprentissage.

Sur-apprentissage : On parle de sur-apprentissage lorsqu'un réseau de neurones s'est trop spécialisé sur ses données d'apprentissages et qu'il a une forte erreur de généralisation.

L'augmentation de données consiste à perturber les images pendant l'apprentissage, avec du bruit, des rotations ou des mises à l'échelle. On peut citer par exemple le *crop* aléatoire, où l'on prend aléatoirement une zone de l'image plutôt que de mettre l'image à l'échelle pour l'entrée du réseau. On utilise également fréquemment des rotations des images, avec des renversement aléatoire de l'image, entre $[-180^\circ, 180^\circ]$. Il est également possible de modifier les contrastes, la teinte ou la saturation, toujours dans le but de déformer les images pour montrer le plus d'exemples différents possible au réseau.

Une forme de régularisation importante et utiliser dans tous les apprentissages aujourd'hui est le "weight decay", ou la normalisation L₂ des poids. Les poids sur les connexions de chaque neurones peuvent prendre des valeurs de forte amplitude pendant l'apprentissage, or cela est pénalisant pour la généralisation, en créant une variance de sortie plus élevée [63, 64]. On souhaite donc forcer les poids à rester petits en pénalisant les grandes valeurs. En utilisant la norme L₂ avec un paramètre α :

$$W = \alpha \sum_{i=1}^n w_i \quad (3.20)$$

On force ainsi les poids à rester dans une sphère de rayon α . Ce paramètre est un *hyperparamètre* à optimiser pendant l'apprentissage, et va dépendre de chaque réseau, avec des valeurs généralement faibles entre 10^{-3} et 10^{-4} .

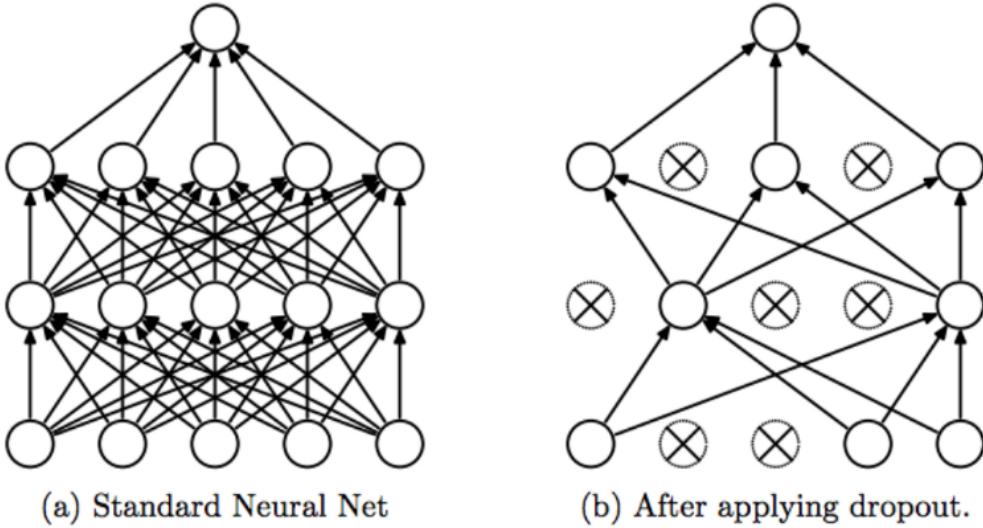


Figure 3.5 : Illustration du *dropout* pris depuis [2]

Une autre forme de régularisation largement utilisée est le *dropout* [2]. Il s'agit de masquer une partie du réseau aléatoirement pendant l'apprentissage. Pour cela, chaque neurone du réseau a une probabilité p d'être abandonné (*dropped*) à chaque passage. La figure 3.5 montre à gauche le réseau initial, et à droite un exemple avec 7 neurones abandonnés, noté X. Une fois l'apprentissage terminé avec ces coupes aléatoires, toutes les connexions sont utilisées lorsque l'on déploie le réseau. Ceci entraîne une activation totale supérieure, car aucune connexion n'est ignorée. Pour garder une activation du même ordre de grandeur que pendant l'entraînement, elles sont toutes réduites d'un facteur p .

Les régularisations que nous venons de présenter ne se font qu'au niveau d'un exemple. Il est toutefois possible d'agir directement au niveau du *batch*, lot de données utilisé pendant l'apprentissage. La normalisation du *batch* (*batch-normalisation*), n'est pas prévue à l'origine pour faire de la régularisation [65]. Utiliser dans les modèles performants récent [66–68], cette méthode n'est pas dédiée à la régularisation, mais améliore effectivement les capacités de généralisation.

NORMALISATION : on désigne par normalisation le fait de centrer-réduire les données (soustraire la moyenne et diviser par l'écart-type). On préfère le terme normalisation ici pour être cohérent l'appellation largement répandue : *batch-normalisation*.

Cette *batch normalization* est utilisée pour accélérer l'apprentissage, en forçant toutes les données à être centrées sur 0. En apprentissage automatique, on centre et réduit généra-

Modèle	Aug.D	W.D.	BatchNorm	P. entrainement	P. test
AlexNet	non	non	NA	100	76.07
AlexNet	non	oui	NA	100	77.36
AlexNet	oui	non	NA	99.82	79.66
AlexNet	oui	oui	NA	99.90	81.22
Inception v3	non	non	non	100	82.00
Inception v3	non	oui	non	100	83.00
Inception v3	non	non	oui	100	85.76
Inception v3	non	oui	oui	100	86.03
Inception v3	oui	non	oui	100	89.31
Inception v3	oui	oui	oui	100	89.05

Table 3.1 : Effets sur AlexNet et Inception de la régularisation [69]

lement les données pour éviter d'avoir à apprendre un décalage en plus de la représentation. L'idée de la normalisation du *batch* est de faire cette opération entre chaque couche, pour que les données soient toujours centrées et réduites. Ceci a deux principales conséquences : on peut utiliser un *learning rate* plus grand car on évite les activations trop grandes ou trop petites, et on limite le sur-apprentissage en ajoutant du bruit entre chaque couche car la variance va dépendre du *batch*. Pour le déploiement du réseau après l'entraînement, la moyenne et l'écart-type utilisés pour la normalisation ne dépendent plus du *batch*, on prend ceux de la population (calculé sur l'ensemble des données d'apprentissage).

On voit sur le tableau 3.1 les effets de chacune des régularisations sur deux réseaux profonds classiques, AlexNet et Inception (cf section 3.2.2) sur la collection d'image ImageNet [56]. On y présente l'effet de l'augmentation de données (Aug. D), du *weight decay* (W.D.) et de la normalisation du *batch* (BatchNorm). Les résultats montrent que, malgré les perturbations apportées par les différentes fonctions, les réseaux parviennent toujours à obtenir des résultats proches de 100% sur le corpus d'apprentissage, en particulier Inception, qui est le réseau le plus performant sur cette tâche. Ceci permet par contre d'améliorer significativement les résultats sur les corpus de test, avec une amélioration de plus de 7% pour Inception et de plus de 5% pour AlexNet. On remarque avec Inception l'apport important de la normalisation du batch, avec de 3.76% à elle seule. On en conclut donc que ces fonctions de régularisations permettent d'améliorer la généralisations des réseaux sur des exemples qu'ils n'ont jamais vu ¹.

¹. Cette affirmation est toutefois à prendre avec précaution, car étant donné la taille des réseaux, il est également possible qu'ils ne fassent que retenir toutes l'informations qu'on leur donne [69].

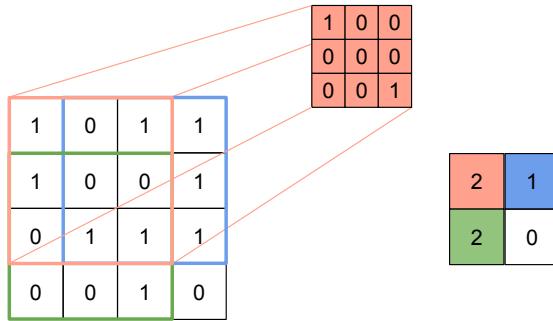


Figure 3.6 : Illustration d'une convolution sur une matrice $4 * 4$.

3.2 RÉSEAUX DE NEURONES CONVOLUTIFS

Nous avons présenté les réseaux de neurones de type perceptrons multi-couches. Notamment en traitement d'image, l'utilisation de convolutions, très répandues, permettent d'obtenir les meilleurs résultats de l'état de l'art [4, 5, 67, 68].

3.2.1 CONVOLUTIONS

Une convolution est une opération entre une matrice M et une matrice de convolution K , appelée **noyau**. L'opération de convolution est définie comme la somme des éléments de M pondérés par les éléments de K . Si K est une matrice carrée de dimension $2n + 1$, alors on peut définir la matrice G résultat de la convolution entre M et K par :

$$G[i, j] = \sum_{u=-n}^n \sum_{v=-n}^n K[u, v] * M[i - u, j - v] \quad (3.21)$$

On peut visualiser cette opération sur la figure 3.6, où une matrice de convolution K de taille $3 * 3$ est appliquée sur une matrice M de taille $5 * 5$. On obtient une matrice G dont la taille va dépendre de la manière dont on applique le noyau sur M . On voit dans l'équation 3.21, que l'on considère que M est toujours définie dans le voisinage de $[i, j]$, ce qui n'est bien entendu pas le cas en pratique, et génère des dépassements de matrice.

On cherche donc à appliquer une matrice de convolution de taille $2n + 1$ sur une matrice (on ne s'intéresse qu'aux convolution de taille impaire). Pour cela on peut définir un pas de décalage entre chaque application. Avoir un pas de 1 signifie que la convolution sera décalée d'un élément après chaque application, et donc qu'elle sera appliquée en chaque position possible de l'image.

On peut ainsi calculer la taille de la sortie de la convolution d'une matrice carrée de

M	K	G

*

Figure 3.7 : Application d'une convolution 3x3 sur une matrice de taille 4x4 avec un padding de 1 et un pas de 1

taille m par une matrice de convolution de taille $s \times s + 1 < m : \frac{m - (2n+1)}{s} + 1$. Elle sera toujours inférieur à m , et donc la convolution va réduire la taille de la matrice d'entrée. Cela vient du fait que la matrice de convolution ne peut pas être appliquée avec pour centre la première ou la dernière ligne ou colonne. Pour corriger ce problème, on peut définir un remplissage (*padding*) autour de la matrice M . En ajoutant ligne et colonne de 0 autour de la matrice M on va pouvoir appliquer la convolution sur un domaine plus grand, sans que cela ne perturbe le résultat, car en ajouter des 0 est sans effet dans l'équation 3.21. Comme on ajoute le padding de chaque côté de la matrice, la taille de la sortie sera : $\frac{m - (2n+1) + 2p}{s} + 1$. En prenant un padding de taille $p = n$, avec un pas de 1, on assure que la taille de la sortie sera égale à la taille de l'entrée (figure 3.7).

Chaque couche d'un réseau convolutif profond va être définie par un ensemble de convolutions, dont la sortie peut être connectée à un neurone. Une image n'étant pas une matrice en deux dimensions, mais un ensemble de matrices, on peut voir celle-ci comme une matrice 3D de taille $L * H * C$ où L et H sont la largeur et la hauteur de l'image, et C le nombre de canaux. Si on applique un certain nombre K de convolutions sur celle-ci, on obtient une matrice 3D de taille $L * H * K$, si la taille d'entrée et de sortie sont identiques, ce qui est illustré par la figure 3.8. En ensemble de convolutions comme celui-ci va définir une **couche de convolution**, composant de base pour un réseau convolutif.

D'autres opérations vont être appliquées pour définir un réseau convolutif, notamment des neurones, généralement de type ReLU, et du *pooling*. L'opération de pooling consiste à mettre en commun (*to pool*) un certain nombre d'élément de la matrice. On prend généralement le maximum (max-pooling) ou la moyenne (average-pooling) d'une région, ce qui

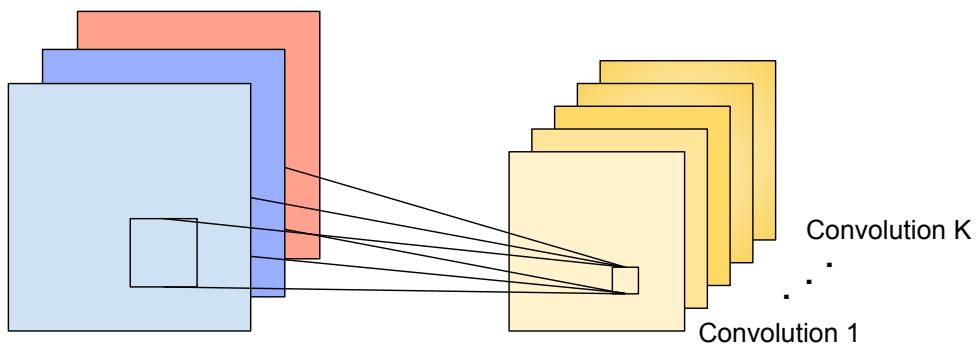


Figure 3.8 : Couche de convolution avec 3 canaux en entrée et 5 en sortie.

permet de faire un sous-échantillonnage, et donc une réduction de la taille de la matrice. En appliquant une pooling $2 * 2$ avec un pas de 2 sur notre matrice $L * H * C$, on obtient la matrice $L/2 * H/2 * C$. La figure 3.9 montre les trois étapes d'une couche d'un réseau convolutif, avec une convolution (fig. 3.9b), des neurones ReLU (fig. 3.9c) et du pooling (fig. 3.9d).

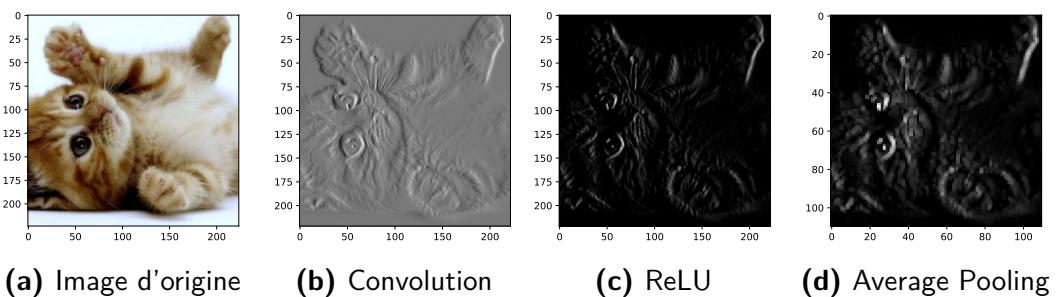


Figure 3.9 : Illustration des opérations de convolution, de ReLU et de *pooling*

3.2.2 RÉSEAUX CONVOLUTIFS PROFONDS

Nous nous intéressons à plusieurs réseaux convolutifs de l'état de l'art qui présentent les éléments les plus importants de tous les réseaux de neurones. L'un des premiers réseaux de neurones convolutifs, et le premier à se répandre, est LeNet présenté par Lecun et al. [3], illustré par la figure 3.10. Ce réseau est constitué de deux couches de convolutions séparées par une couche de sous-échantillonnage (similaire à l'*average-pooling*). Une partie entièrement connectée permet de passer d'une sortie de taille $16 * 5 * 5$ à une sortie de dimension

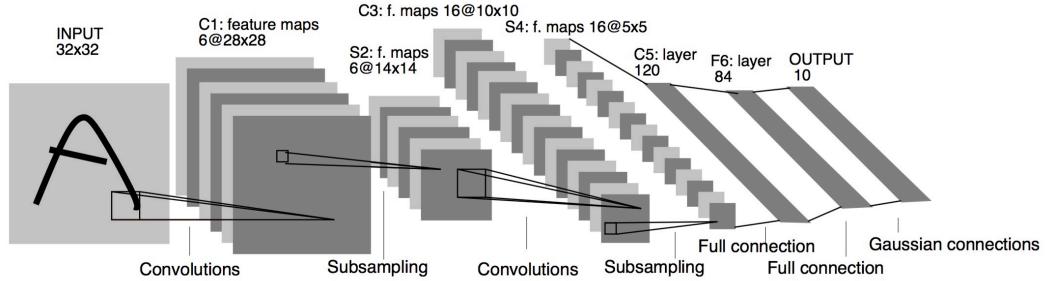


Figure 3.10 : Schéma du réseau profond LeNet [3]

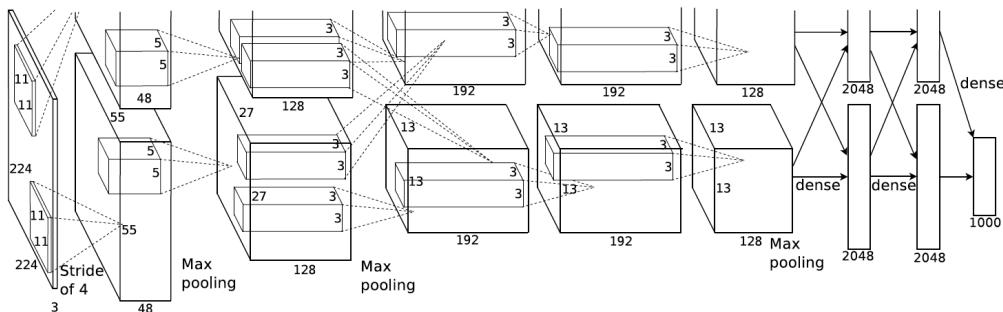


Figure 3.11 : Architecture AlexNet [4]

10, correspondant aux dix classes MNIST [3], la tâche de reconnaissance de chiffre pour laquelle le réseau a été développé.

Ce réseau dispose de toutes les briques nécessaires pour la construction de tous les réseaux convolutifs qui suivront. Notamment avec une partie “caractéristiques”, à savoir un ensemble de convolutions, qui sont chargées d’extraire les caractéristiques visuelles. Et une partie “classification”, avec des couches entièrement connectées, chargées de réaliser la classification en 10 catégories dans le cas présent.

L’architecture AlexNet, présentée en 2012 [4] pour la tâche de classification ImageNet [56]², utilise une approche similaire. Bien que cette architecture soit largement plus grande, avec plus 60 millions de paramètres, l’organisation est proche de celle de LeNet (figure 3.11), avec une partie extraction de caractéristiques, composée de cinq convolutions, et une partie classification composée de couches entièrement connectées.

Ce réseau utilise des convolutions de 11×11 et de 3×3 , chacune séparée par un max-pooling de taille 3×3 , ce qui représente un grand nombre de sous-échantillonnage. Une

². La collection d’image ImageNet est très utilisée pour l’apprentissage des réseaux de neurones, car elle dispose de plusieurs millions d’images annotées, et répartit en plusieurs milliers de classes.

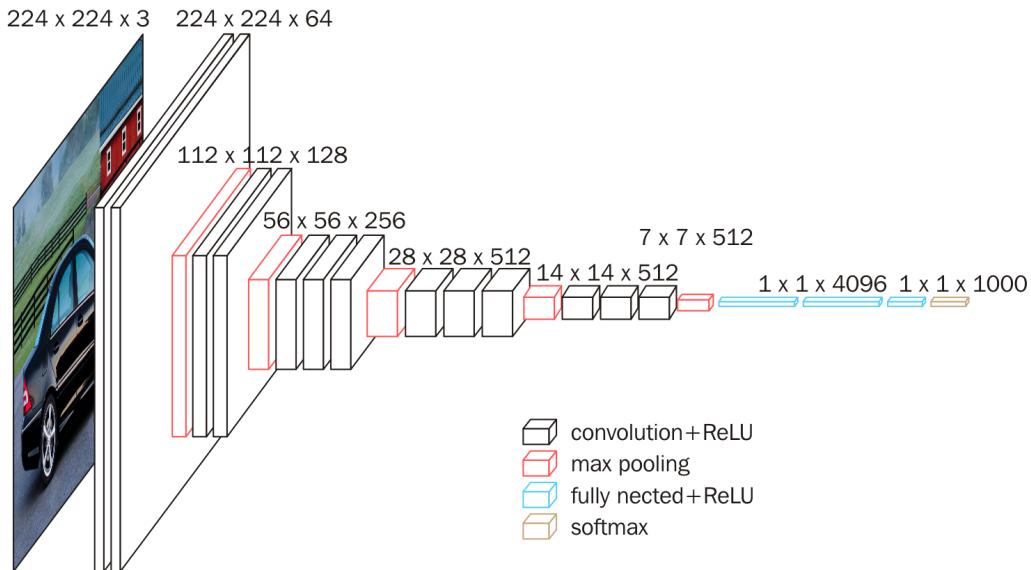


Figure 3.12 : Architecture VGG [5]

des nouveautés de ce réseau est l'introduction des dropout (présentés section 3.1.5), qui lui permettent de ne pas sur-apprendre, malgré sa grande taille. Cette nouvelle architecture atteint 18.2% d'erreur ImageNet³, ce qui est bien inférieur aux résultats précédent sans réseaux de neurones, qui étaient de 26.2% avec les descripteurs visuels SIFT (Scale-Invariant Feature Transform) [70].

Ces résultats sur un problème de vision bien connu ont permis au réseaux de neurones d'être à nouveau au centre de l'attention de la communauté scientifique. Les nouveautés proposées par Simonyan et Zisserman [5] avec leur réseau VGG, ont posé de nouvelles bases pour la constructions de réseaux profonds. Premièrement, ils proposent de n'utiliser que des convolutions 3×3 . En utilisant des convolutions de petites tailles, on peut augmenter la profondeur du réseau, ce qui en pratique donne de meilleurs résultats. Le réseau VGG présenté (figure 3.12) est plus profond, il utilise ici 16 couches, dont 13 de convolutions et 3 entièrement connectés pour la classification. Une version avec 19 couches est également proposée. Ceci fait passer le nombre de paramètres à 144 millions, et on commence alors à parler de réseaux très profonds.

Les convolutions 3×3 ne sont pas toutes séparées par du sous-échantillonnage comme dans les réseaux précédent. En effet, deux convolution 3×3 à la suite ont le même champ récepteur qu'une convolution 5×5 , s'il n'y a pas de pooling entre les deux, et 3×3 à la suite ont

³. Ces résultats, et tout ceux qui suivent, sont basés sur une réimplémentation des réseaux, et peuvent différer de ceux des publications originales. Nous nous intéressons surtout à l'ordre de grandeur et aux écarts entre les résultats, qui eux ne changent pas.

le même qu'une convolution $7 * 7$. Remplacer une convolution $7 * 7$ par trois convolution $3 * 3$ permet de multiplier par trois le nombre d'activations non-linéaires ReLU, ce qui rend la fonction plus discriminante, en plus de réduire considérablement le nombre de paramètres (81% de paramètres en moins). On obtient alors un réseau plus profond, et plus performant, arrivant à un taux d'erreur de 8.1% sur ImageNet [5].

En plus de se limiter à des convolutions $3 * 3$, VGG propose d'utiliser deux types de couches dans le réseau. Premièrement des couches qui conserve la taille des entrées, donc avec un pas de 1 et un *padding* de 1, et des couches qui réduise la taille par deux, avec un pas de 2, mais qui multiplient par deux le nombre de canaux. En enchaînant ces deux types de couches, on peut construire un réseau de plus en plus profond, avec moins de disparition du gradient.

Le problème de l'approfondissement des réseaux est la diminution du gradient au fur et à mesure de la rétro-propagation. En effet, à chaque couche, le gradient va diminuer légèrement. Alors que ce n'est pas un problème pour des réseaux avec quelques couches, si l'on veut continuer à approfondir les réseaux, il réduire l'impact de cette diminution. L'architecture de réseau ResNet [6] propose un nouveau type de connexion : les *skip-connections*, ou plus précisément les connexions résiduelles. Pour éviter que chaque couche fasse disparaître petit à petit le gradient, on ajoute une connexion de l'entrée de la convolution directement vers la sortie. Cela crée deux "chemins" pour la propagation et surtout la rétro-propagation. Un exemple est montré sur la figure 3.13, avec un bloc résiduel, constitué de deux couches de convolutions et d'une connexion résiduelle entre l'entrée et la sortie. Ici la fusion est faite avec une addition, ce qui montre les meilleurs résultats en pratique, car permet une meilleure propagation de l'information [71].

Bloc : avec la taille grandissante des réseaux de neurones, le terme blocs de convolutions a été introduit. Un bloc est un ensemble d'un certain nombre de couches d'un réseau. Cela permet de définir plus simplement des réseaux très profonds, quand certaines structures se répètent dans le réseau.

Ces connexions résiduelles permettent d'agrandir grandement les réseaux, ainsi l'architecture ResNet présente les meilleurs résultats avec une organisation en 200 couches, avec une erreur sur ImageNet de 4.8% [71]. Les *skip-connection* peuvent également être utilisées pour connecter le réseau plus densément. L'architecture DenseNet [48] propose un bloc de base où chaque couche précédente est connectée aux couches suivantes. Ces nombreuses nouvelles connections ont plusieurs avantages, notamment de limiter la dispari-

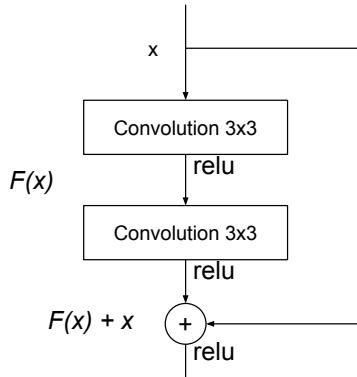


Figure 3.13 : Exemple d'un bloc résiduel de ResNet (source [6]).

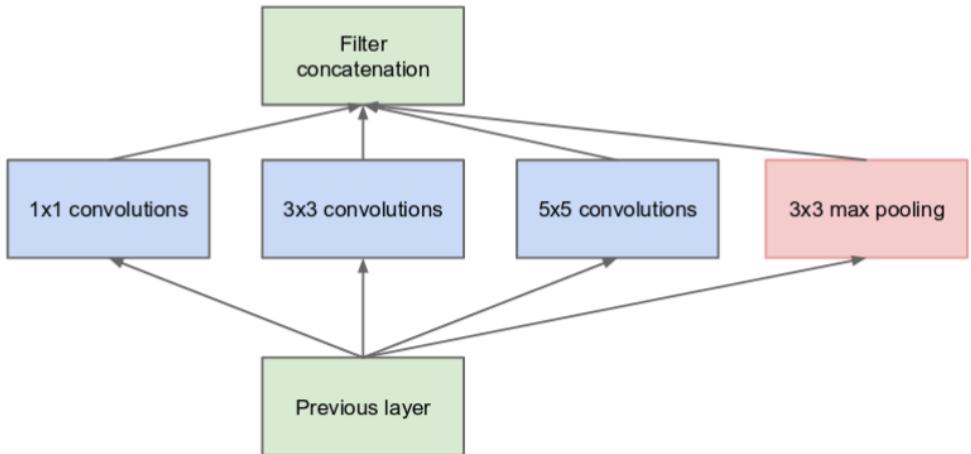
tion du gradient, de renforcer la propagation des caractéristiques extraites, et de permettre la réutilisation des caractéristiques extraites précédemment dans le réseau.

Augmenter la taille du réseau ne se fait pas uniquement en profondeur, mais également en largeur. On peut, dans un bloc, également opérer à plusieurs échelles, en utilisant des convolutions 1×1 , 3×3 et 5×5 sur une seule couche. C'est le cas de l'architecture Inception [7] qui, pour chaque couche, va multiplier le nombre de convolutions parallèles. Les architectures suivantes [66], jusqu'à InceptionV4 [72], reposent sur la même idée d'augmenter la largeur des réseaux, pour travailler à plusieurs échelles sur chaque couche. Ce type de réseaux a permis de réduire l'erreur sur ImageNet à 3.1% [72].

Avec l'agrandissement des réseaux, en profondeur et en largeur, la précision sur des tâches de classifications précises comme ImageNet s'améliore [7, 66, 72]. Cependant, cela mène à des réseaux de plus en plus grands, en terme de mémoire, et de plus en plus lents. Dans le cas d'une application en mobilité, il n'est pas envisageable d'explorer des réseaux encore plus profonds. Les architectures présentées précédemment mettent en avant les caractéristiques principales qu'un réseau doit avoir pour obtenir de bon résultats, mais il nous faut viser une diminution de la taille et l'accélération pour garantir des traitements compatibles avec l'interaction d'un humain.

3.3 RÉDUCTION DE LA TAILLE ET DE LA COMPLEXITÉ DES RÉSEAUX

Nous avons présenté des réseaux très profonds, avec de plus en plus de paramètres, pour obtenir des résultats toujours meilleurs. Le problème étant qu'avec la taille grandissante des réseaux, leur utilisation sur des processeurs mobiles par exemple devient difficile. Nous nous intéressons dans cette section à des méthodes de réduction de la taille des réseaux,



(a) Inception module, naïve version

Figure 3.14 : Bloc Inception (source [7]).

soit par compression, soit en repensant l’architecture, pour que la taille et la rapidité des réseaux deviennent un objectif.

3.3.1 COMPRESSION DES RÉSEAUX

Une première approche pour la réduction de la taille des réseaux est la suppression des connexions “inutiles”. Appelé *pruning*, elle consiste à identifier les connexions inutiles pour les supprimer. Au début des années 90, cette méthode a beaucoup été étudiée pour réduire la complexité afin de faciliter l’apprentissage [47, 63, 73], notamment à cause des limitations techniques de l’époque. Il est possible d’appliquer le *pruning* à des réseaux convolutifs en apprenant quelles connexions sont nécessaires [74, 75]. Les connexions avec des poids faibles, en dessous d’un certain seuil, sont supprimées, et le réseau est ré-entraîné, et ainsi de suite. Cette approche permet de réduire considérablement le nombre de paramètres, avec par exemple un réduction d’un facteur 9 sur AlexNet et d’un facteur 13 sur VGG.

Il est également possible de réduire la taille en mémoire d’un réseau en réduisant le nombre de bits nécessaires à la représentation de chaque poids. On peut regrouper les poids à l’aide d’une fonction de hachage [76], qui se fait avant l’apprentissage. En utilisant un algorithme de regroupement, comme k-means [77], il est possible de regrouper les poids similaires en un certain nombre de groupes, une fois l’apprentissage réalisé. Ceci permet de garder les performances du réseau d’origine, et de tirer profit de la redondance dans les réseaux. La binarisation totale du réseau permet non seulement de grandement réduire la taille du réseau, jusqu’à 32 fois, mais permet surtout l’accélération des calculs,

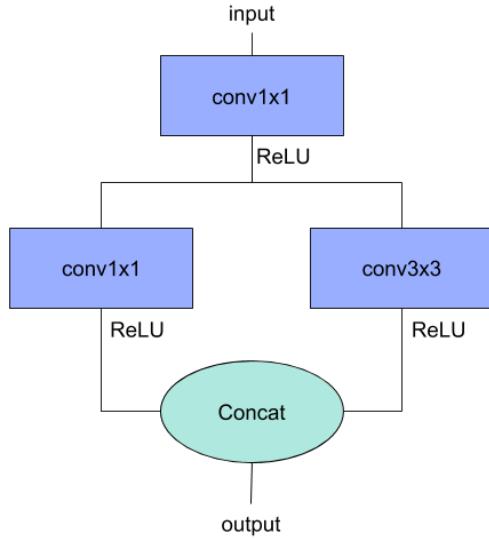


Figure 3.15 : Module “FIRE” de SqueezeNet [8]

jusqu’à 60 fois [78]. Ceci entraîne cependant une diminution logique de la qualité des résultats.

3.3.2 CONVOLUTIONS 1×1

Introduit par SqueezeNet [8], les réseaux convolutifs avec *micro-architecture* représentent une autre manière de créer des réseaux de tailles réduites. En créant un réseau de petite taille dès le départ, on peut apprendre le réseau de bout en bout avec moins de paramètres. Grâce aux dernières avancées sur les réseaux de neurones, notamment les convolutions 1×1 et les *skip-connections*, on est capable de créer des architectures avec un nombre de paramètres largement réduit, qui arrive à des résultats proches de AlexNet ou VGG [8, 20].

La première stratégie pour réduire le nombre de paramètres est l’utilisation de convolution 1×1 . On peut voir ce type de convolution comme une fusion des informations au niveau des pixels, car une convolution regardera le même pixel sur l’ensemble des canaux. Ce type de convolution utilise logiquement 9 fois moins de paramètres qu’une convolution 3×3 . Quelque soit le type de convolution, il est également utile de s’intéresser aux nombres de canaux d’entrée, car ils vont déterminer le nombre de paramètres de la couche. En combinant des convolutions 1×1 et des convolutions 3×3 avec le moins de canaux possibles en entrée, SqueezeNet définit les *fire-modules* (figure 3.15, un type de bloc de convolutions utilisant peu de paramètres).

Dans le bloc *fire*, la première couche nommée *squeeze* réduit le nombre de canaux pour les convolutions 3×3 de la deuxième couches pour limiter le nombres de paramètres. Ce

bloc ne réalise pas de sous-échantillonnage, la taille de l'entrée et de la sortie sont donc identiques, seul le nombre de canal va augmenter. Le sous-échantillonnage est réalisé par des couches de *maxpooling*, assez loin⁴ dans le réseaux, ce qui a tendance à donner de meilleurs résultats, en augmentant la valeur des cartes d'activations [6]. En appliquant ces différentes stratégies, SqueezeNet a des résultats comparables à ceux de AlexNet sur ImageNet, avec un modèle environ 50 fois plus petit.

3.3.3 CONVOLUTIONS GROUPÉES

Les convolutions groupées [9] ont d'abord été introduites pour permettre de distribuer les calculs pendant l'apprentissage du réseau [4]. On sépare les canaux en un certain nombre g de groupes, et les convolutions ne vont être utilisées que sur un groupe particulier. Si l'on a C canaux d'entrée avec g groupes, on a une entrée qui va être composée de groupes de taille C/g , où chaque convolution ne va être appliquée que sur un seul de ces groupes. Comme illustré sur la figure 3.16, avec un nombre de groupes égal à deux, on divise les convolutions en deux blocs, celle du bloc 1 n'étant effectuée que sur les canaux numérotés $[1; C/g]$ n'ayant pas accès aux canaux $]C/g; C]$

Les convolutions groupées peuvent être apprises séparément (notamment sur plusieurs GPUs) et présentent l'avantage de réduire le nombre de paramètres, avec moins de connexions. Ces convolutions groupées peuvent être étendues : les *DeepWise Convolution* [20] consiste à prendre un nombre de groupes égale au nombre de canaux en entrée. Ainsi, chaque canal de sortie n'est connecté qu'à un seul canal d'entrée, comme illustré par la figure 3.17. Le nombre de paramètres est alors grandement réduit, d'un facteur C , mais on perd en contrepartie toute forme d'information entre les canaux. Pour cela, MobileNet [20] définit un bloc composé d'une convolution *deepwise* 3×3 et d'une convolution 1×1 chargé de combiner les informations venant des de chacun des canaux de la convolution *deepwise*. Ce bloc contient moins de paramètres qu'une convolution 3×3 classique, et obtient des résultats proches de ceux de VGG [20], avec en plus l'ajout d'un couche de non-linéarité possible entre les deux convolutions.

L'architecture ShuffleNet [10] tire profit des deux approches précédentes. Elle utilise un type de bloc proche de celui de Mobile, mais ajoute une convolution 1×1 groupée pour réduire le nombre de canaux en entrée de la convolution 3×3 . La convolution 1×1 de fusion des informations est remplacée par une convolution 1×1 groupée (figure 3.18). Mais pour que celle-ci ait quand même un rôle de fusion d'information, une opération de mélange

4. Par "loin" on entend ici "proche de la sortie".

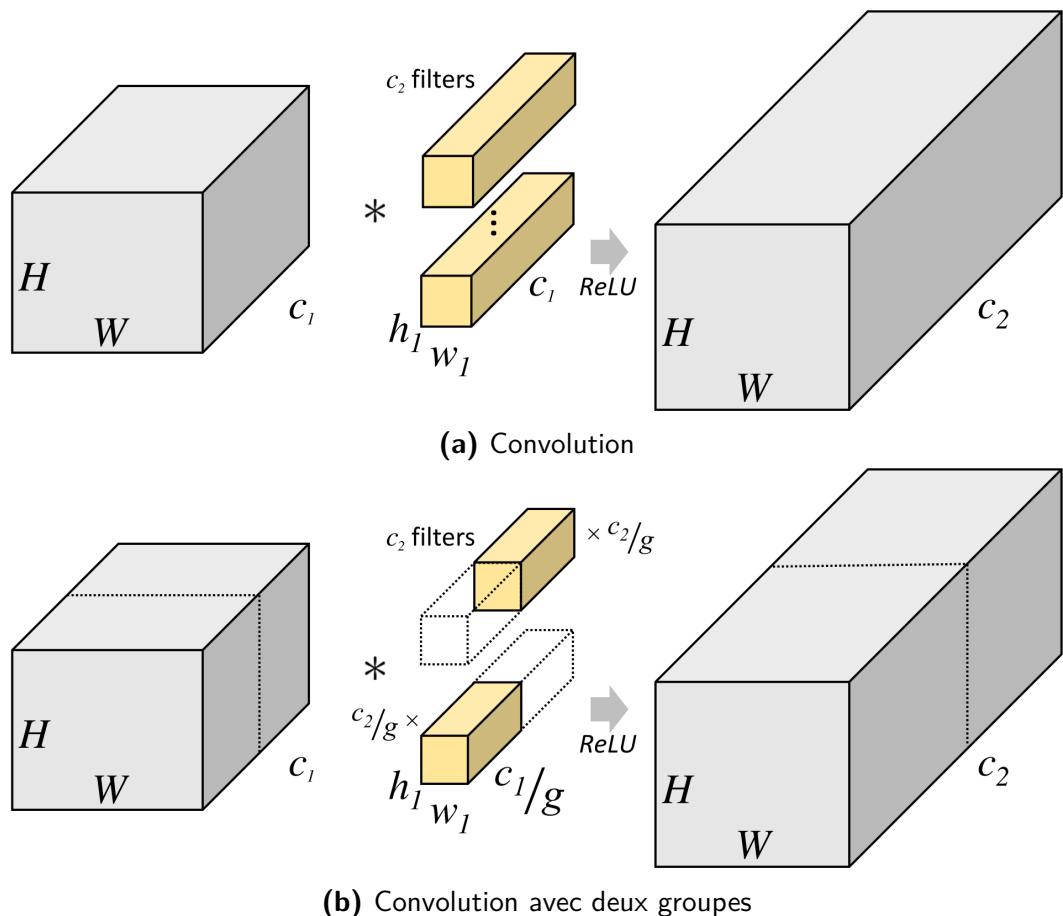


Figure 3.16 : Illustration des convolutions groupées (source [9])

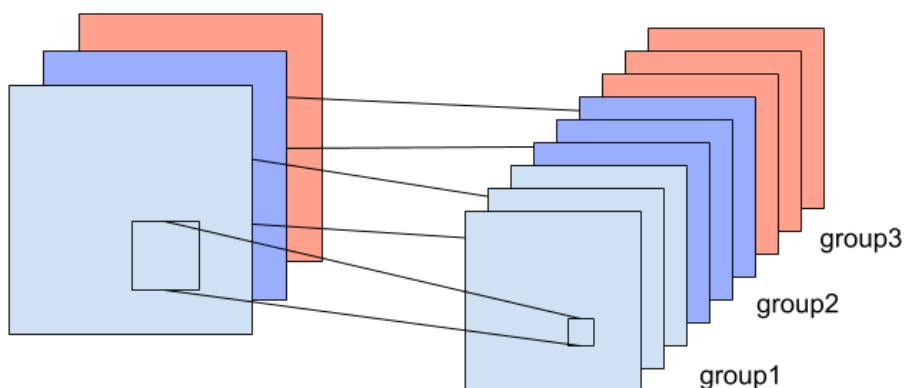


Figure 3.17 : Exemple de *DeepWise Convolution*), avec trois groupes pour trois canaux d'entrée.

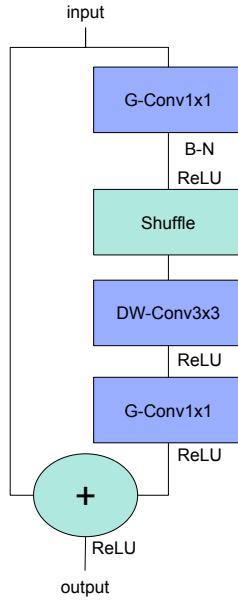


Figure 3.18 : Module “Shuffle” de ShuffleNet [10]

des canaux est réalisée entre les deux convolutions, en plus d’ajouter une *skip-connection*. Cette approche permet d’obtenir des résultats légèrement supérieurs à MobileNet, mais avec environ 4.5 fois moins de paramètres grâce aux convolutions groupées.

Ces différentes micro-architectures permettent d’obtenir des résultats proches ou parfois supérieurs aux réseaux profonds avec des millions de paramètres sur des tâches comme ImageNet. Ils montrent surtout différentes stratégies de réduction des architectures intéressantes. Dans le chapitre 6, nous nous intéressons à un réseau de petite taille adapté à une application en mobilité, et utilisons ces différentes méthodes pour construire un réseau adapté à la reconnaissance de gestes.

3.4 RÉSEAUX AVEC DÉTECTION DE RÉGIONS

L’utilisation des réseaux de neurones convolutifs n’est pas limitée à la classification des images, et des méta-architecture permettent par exemple leur utilisations pour la détection des régions d’intérêt sur les images. Par méta-architecture, nous entendons une architecture construite autour d’une architecture précédemment définie, comme celles que nous avons présentées dans la section 3.2.2. En utilisant un réseau appris à reconnaître un certain nombre d’objet, il est possible détecter où cet objet se situe dans l’image. Pour cela on distingue deux types d’approches, avec des réseaux à proposition de régions (section 3.4.1), ou à l’aide de carte d’activation (section 3.4.2).

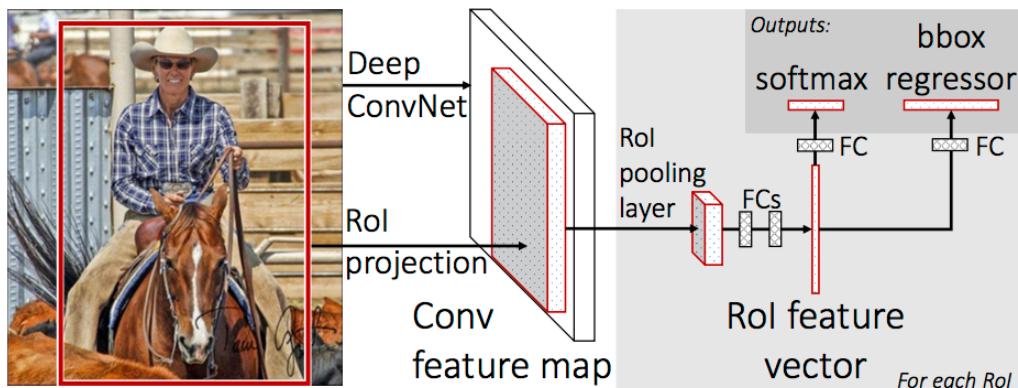


Figure 3.19 : Architecture Fast-RCNN [11]

3.4.1 PROPOSITION DE RÉGIONS

Il est possible de faire apprendre directement à un réseau les régions d'intérêt sur les images, si l'on dispose d'un corpus d'apprentissage avec les régions annotées. Des collections d'images comme MS COCO [79] proposent un grand nombre d'images avec des boîtes englobantes autour des objets. Ceci permet de construire un réseau qui va à la fois classifier image et proposer une région pour la localisation de l'objet. Les réseaux de type R-CNN (pour Region Convolutional Neural Network) [80] permettent de créer de proposer des boîtes englobantes pour les objets présents dans l'image. La proposition Fast-RCNN [11], présentée sur la figure 3.19, consiste à permettre au réseau de produire deux sorties, une de classification, et une de région d'intérêt. Le réseau apprend à produire la boîte englobante correspondant à celle donnée par le corpus d'apprentissage, et à classifier uniquement cette zone.

L'approche Faster-RCNN [50] ajoute un *Region Proposal Network* après la dernière couche de convolution. Ce réseau est capable de regarder les caractéristiques de la dernière couche de convolution pour déterminer une région d'intérêt probable. Ensuite, la même méthode de classification que cette de Fast R-CNN

3.4.2 CARTES D'ACTIVATIONS

Il a été montré que les réseaux convolutifs étaient capables de détecter les objets dans une images même si l'apprentissage s'est fait sur des images entières, sans indications sur la présence des objets. Avec un réseau entraîné sur ImageNet, en regardant où se situe les maximum d'activation dans le différents filtres de convolutions, on remarque ceux-ci se concentre sur l'objet à reconnaître [81]. Mais surtout, entraîné à reconnaître des scènes,

comme des pièces ou des lieux extérieurs, on note des activations sur des objets bien particuliers (lampes, lits, etc) alors qu'aucune information sur ces objets n'a été fournie au réseau [82]. Ceci sous entend que le réseau a bien une représentation interne de ces objets, lui permettant de les reconnaître, et de reconnaître les scènes. Toutefois, ces informations de localisation sont perdues avec l'utilisation de couches entièrement connectées. En effet, ces couches utilisées pour la classification fusionnent les informations venant de tous les canaux, et la localisation des objets devient impossible. Il est cependant possible d'extraire l'information avant qu'elle ne passe à travers les couches entièrement connectées, pour ainsi avoir deux formes de sorties, une qui indique la position de l'objet, et l'autre qui indique sa classe [83]. Néanmoins, ceci ne permet pas de faire un apprentissage de ces régions et les zones d'activations ne correspondent pas forcément à ce qui est classifié, comme on ne tient pas compte de l'information présente dans les couches de classification pour savoir ce qui est important où non pour telle ou telle classe.

Un autre manière d'obtenir une carte d'activation est d'utiliser une image plus grande, de la séparer en sous-régions, et de classifier chacune des régions. En plus d'être très coûteuse, car il faut appliquer le réseau une fois pour chaque pixel de la carte d'activation, un réseau composé uniquement de convolutions réalise la même opération. Les **réseaux entièrement convolutifs**, c'est-à-dire sans aucune couche entièrement connectée, présentent deux principaux avantages : ils permettent de garder des informations sur l'organisation spatiale de l'image sur l'ensemble du réseau, et ils peuvent être appliqués facilement sur n'importe quelle taille d'image étant donné qu'ils ne sont composé que de convolutions. Il est possible de transformer une couche entièrement connectée en une convolution 1×1 , comme le type d'opération est similaire, avec les mêmes paramètres [12].

La sortie d'un réseau entièrement convolutif représente, comme dans un réseau classique, la probabilité de classification pour chaque classe. Cependant, un réseau de neurones à un **champ de réception** limité, c'est-à-dire qu'il ne regarde que sur une image d'une dimension donnée. En appliquant un réseau entièrement convolutifs, on peut l'appliquer sur une image de dimension arbitraire. Ainsi, plus l'image d'entrée est grande, plus la sortie est grande. On peut ainsi interpréter la sortie comme une carte d'activation du réseau sur l'image (figure 3.20).

En ajoutant des connexions entre l'entrée et la sortie, ou en utilisant une base de données avec des informations sur les régions d'intérêt, il est possible d'apprendre plus précisément cette carte d'activation [12]. Même si ces méthodes ne permettent pas d'obtenir des régions d'intérêt, mais uniquement des zones d'activation, elles ont l'avantage de ne pas nécessiter d'apprentissage particulier sur des corpus avec des régions annotées. C'est un point impor-

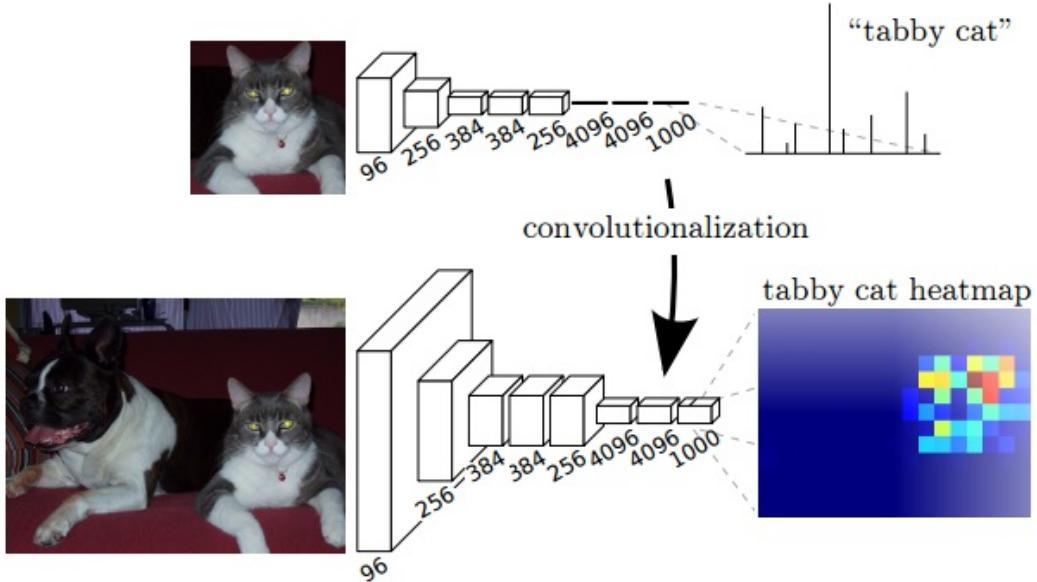


Figure 3.20 : Transformation d'une couche entièrement connectée en convolution pour obtenir une carte d'activation [12]

tant dans notre cas, et dans le chapitre 5, nous utilisons ces réseaux entièrement connectés pour nous aider dans la détection des œuvres.

3.5 IDENTIFICATION D'INSTANCES

La recherche d'instances est une tâche de recherche visuelle qui vise, à une image de requête, à récupérer toutes les images qui contiennent la même instance d'objet dans une base de données d'images. La recherche d'instances a une large gamme d'applications, comme par exemple la recherche inversée d'image sur le web ou l'organisation de collections de photos personnelles. Dans notre cas, nous utilisons la recherche d'instances pour réaliser l'identification d'œuvres avec le dispositif GUIMUTEIC.

Malgré les avancées récentes des réseaux de neurones que nous avons présentées précédemment, ils sont resté longtemps en deçà des méthodes à bases de descripteurs visuels pour la recherche d'instances [84, 85]. Les approches modernes basées sur l'apprentissage de similarité et l'apprentissage de classement (learning to rank) permettent d'obtenir les meilleurs résultats aujourd'hui [86].

3.5.1 CARACTÉRISTIQUES INGÉNIÉRÉES

La recherche de similarité entre les images a longtemps été basée sur la correspondance entre les descripteurs visuels extraits des images. Un descripteur visuel capture une carac-

téristique d'un point de l'image. Dans le cas idéal, cette description doit être invariante à l'orientation, à l'échelle, aux différences de prise de vue, etc. Les méthodes d'extraction de descripteurs se basent sur la détection de points clefs de l'image, l'extraction de caractéristique de chacun de ces points, pour finalement représenter l'image par l'ensemble de ces descripteurs. La qualité de la représentation de ces images va être dépendante de deux éléments principaux : si les points clefs représentent bien l'image et si l'extraction de caractéristique permet de capturer suffisamment d'information.

Le descripteur SIFT (Scale Invariant Feature Transform) développé par Lowe [87] présente de très bon résultats notamment pour la reconnaissance d'objet [88–91] ou la recherche d'image [92]. SIFT combine un détecteur de régions invariant à échelle (par différence de gaussiennes) et un descripteur basé sur la distribution de gradient dans les régions détectées. Cependant, il nécessite une grande complexité de calcul, ce qui est un inconvénient dans notre cas, et en général dans les applications en temps réel [88, 93].

De nombreuses variantes à SIFT existent pour régler certains problèmes, notamment la vitesse de calcul [94], ou la précision du descripteur [95]. Speed Up robust Features (SURF) est une approximation de SIFT, qui est plus rapide, sans réduire la qualité des points détectés [96]. Les descripteurs BRIEF (Robust Indépendant Elementary Features) proposent une alternative aux descripteurs de SIFT, beaucoup moins complexe, avec une précision d'appariement entre les descripteurs très proche [97]. Dans le but de viser les appareils mobiles, le descripteur ORB (Oriented FAST and Rotated BRIEF) [98] couple le détecteur de points clefs FAST [99], qui est plus rapide que les différences de gaussienne de SIFT, avec le descripteur BRIEF. Ce dernier descripteur permet d'avoir une solution utilisable sur processeur mobile, avec des performances proches de celle de SIFT [98].

Dans le cadre de la recherche d'image, ou l'identification d'instances, la plupart des méthodes à base de descripteurs visuels utilisent le même type d'approche : l'image est représentée par un certain nombre de descripteurs, qui sont regroupés en un vecteur ou un ensemble de vecteurs, et on compare l'image requête à l'ensemble des images de la base de données grâce à ces vecteurs. A la manière des sacs de mots en recherche d'information textuelle, on peut regrouper ces descripteurs en sac de mots visuels [100], ce qui les rend compatible avec les fichiers inverses [101], avec des représentations binaires ou compactes [102–104], avec les Fisher Vector [105], les VLAD [106] ou avec l'ajout d'extension de requêtes [107, 108]. Les descripteurs visuels permettent également une vérification géométrique pour la vérification des résultats de correspondance entre les descripteurs [103, 109]. Cependant, les solutions à base de réseaux de neurones présentent aujourd'hui les meilleurs résultats et proposent de nombreuses solutions adaptées [13].

3.5.2 EXTRACTION DE CARACTÉRISTIQUES

Il a été présenté par Krizhevsky et al. [4] qu'extraire des caractéristiques de l'image depuis les couches cachées d'un réseau de neurones pourrait remplacer les descripteurs ingénierés, ce qui sera prouvé plus tard par de bons résultats pour la recherche d'instances [110]. Il est en effet possible d'utiliser tout ou une partie d'un réseau de neurones sur une tâche différente de laquelle il a été entraîné. Comme montré dans les sections précédentes, un CNN typique se compose de plusieurs couches convolutionnelles, suivies de couches entièrement connectées et se termine par une couche SoftMax produisant une distribution de probabilité sur les différentes classes. Au lieu d'utiliser ce classificateur intégré, il est possible de considérer les activations des couches intermédiaires comme représentation d'image, comme montré sur la figure 3.21. Il est ainsi possible d'utiliser un autre classifieur sur cette nouvelle sortie du réseau, sur une tâche complètement différente de celle de départ, avec pourtant de bons résultats [111].

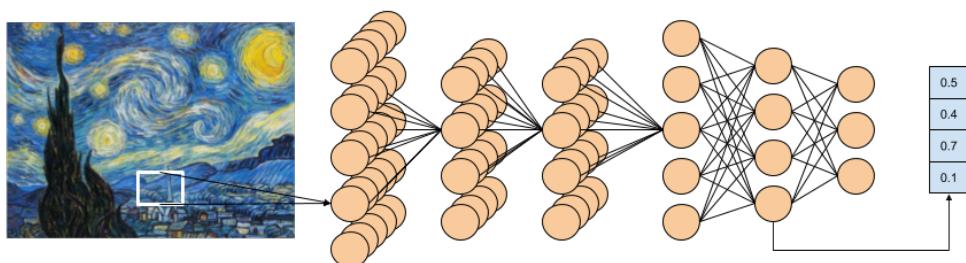


Figure 3.21 : Exemple d'extraction de caractéristiques depuis une couche cachée d'un réseau de neurones.

Ces représentations d'images "apprises"⁵ permettent d'obtenir des résultats parfois meilleurs que les descripteurs ingénierés [110]. Cependant, elles souffrent d'un manque de robustesse à l'orientation, à la mise à l'échelle ou au bruit, ainsi que de l'impossibilité de les régler précisément pour une tâche, les rendent moins performants. Les travaux récents montrent que l'utilisation des sorties des couches intermédiaires plutôt que des dernières couches entièrement connectées donnent de meilleurs résultats. Particulièrement lorsque l'on s'in-

⁵. On parle de représentation d'image apprise car elle est issue d'un réseau de neurones entraîné. Elle n'est cependant pas forcément adapté à la tâche sur laquelle elle est utilisé.

téresse aux objets dans les images, notamment grâce à une plus grande généralité des caractéristiques, moins spécialisées sur la tâche apprise [112, 113].

Plusieurs améliorations ont été proposées pour palier à leur manque de robustesse, notamment à base de régions sur les images. En utilisant les sorties du réseau de neurones sur un certain nombre de régions de l'image, on obtient une description des régions, qu'il est possible d'accumuler [114–117], ou d'agréger en Fisher Vector [118] ou avec des VLAD [119, 120]. L'avantage de l'utilisation des régions est qu'une vérification géométrique est possible lorsque l'on compare deux images, et que le ratio des images peut être conserver, ce qui évite les déformations [86, 117]. Cependant, l'utilisation de caractéristiques issues d'un réseau entraîné sur une collection standard, généralement ImageNet, ne peut pas garantir qu'elles soient adaptées à la tâche qui nous intéresse.

Lorsque l'on travaille sur des collections de recherche d'images, il n'est généralement pas possible d'entraîner un réseau de neurones profonds complet sur celles-ci, car il y a un nombre de classes très grand pour un nombre d'exemples très limité, en plus d'une faible variabilité entre les images d'exemple. La plupart des méthodes présentées précédemment utilisent un réseau pré-entraîné sur ImageNet ou autre grande collection. On peut réaliser un réaliser un fine-tuning (voir encadré et figure 3.22) sur la collection de recherche d'image, et malgré des résultats qui ne sont généralement pas suffisants pour permettre une bonne classification des images, ce fine-tuning améliore les résultats de l'extraction de caractéristiques [86, 121, 122]

Fine-tuning : On entend par fine-tuning (ou apprentissage fin) le fait de remplacer la dernière couche de classification d'un réseau appris sur une tâche donnée par une nouvelle et d'entraîner tout ou un partie du réseau sur cette nouvelle tâche. Cette méthode est généralement utilisée lorsque l'on doit travailler sur des collections insuffisamment grandes ou variées pour qu'un réseau profond soit appris.

La figure 3.23, tirée de [13], retrace la chronologie des recherches présentées précédemment. On remarque l'entrée dans l'ère du CNN, pour citer l'auteur, avec d'abord l'extraction de caractéristiques globales depuis un réseau pré-traîné, jusqu'à la représentation de régions par Tolias et al. [117]. Cette chronologie s'arrête en 2016, avec un nouveau changement de paradigme, qui est l'arrivé des réseaux siamois, notamment avec l'arrivé de FaceNet [15].

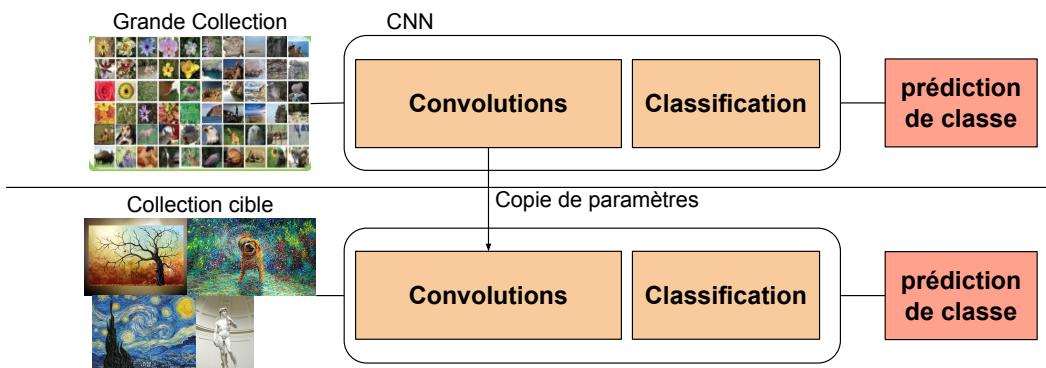


Figure 3.22 : Illustration du fine-tuning, avec apprentissage sur première collection et second apprentissage sur la collection cible.

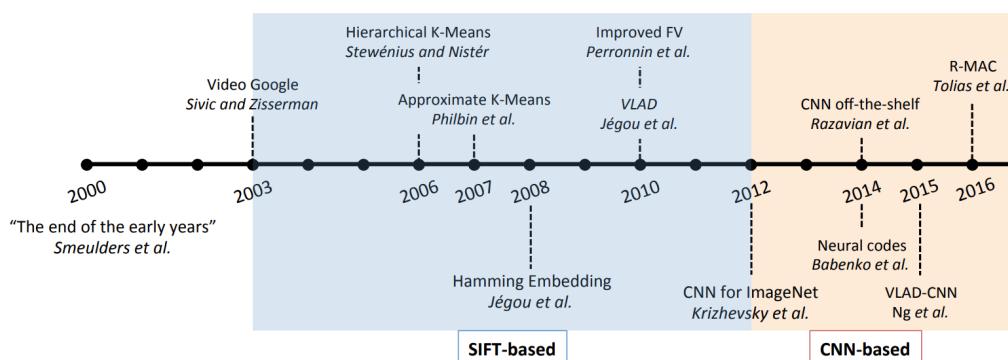


Figure 3.23 : Chronologie sélective de la recherche d'instance (source [13]).

3.5.3 APPRENTISSAGE DE SIMILARITÉ

Les réseaux siamois [14, 123] ont été introduits pour résoudre le problème de comparaison entre deux images. Le réseau proposé par [14] est montré sur la figure 3.24. Il s'agit d'utiliser deux fois le même réseau, avec deux entrées différentes. On présente alors au réseau un ensemble de couples, des paires d'images représentant le même objet ou non. Les deux sorties du réseau sont alors comparé, et l'apprentissage se fait par rétro-propagation [47] sur les deux branches.

L'avantage de ce type de méthode est que l'apprentissage correspond à la tâche pour laquelle le réseau est utilisé, contrairement aux approches de la section précédente. Il est alors possible d'entraîner un réseau à produire une projection des images qui correspond à la problématique qui nous intéresse. Par exemple, si la taille de sortie du réseau est de dimension N , et que l'on compare les sorties de chacune des deux branches du réseau avec une distance (euclidienne ou autre), il est possible d'apprendre au réseau à projeter les

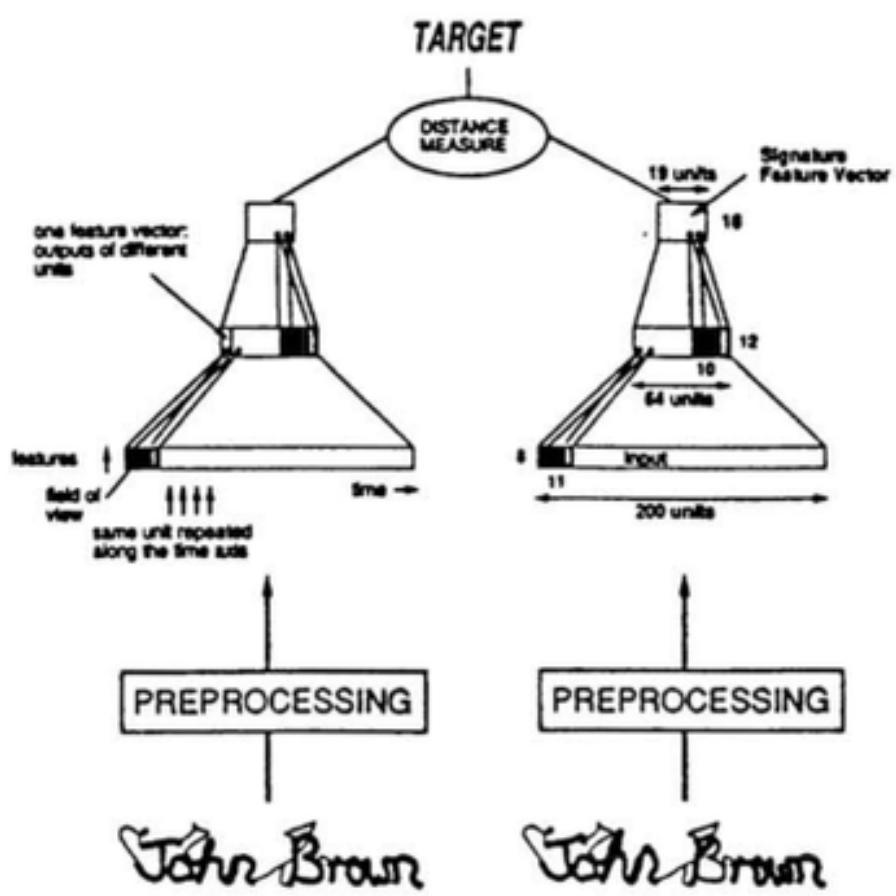


Figure 3.24 : Réseau siamois présenté par [14]

images dans cet espace de dimension N pour que la distance entre les projections représente la similarité entre les images. Soient I_1 et I_2 deux images, et \hat{x} qui vaut 0 si les images représentent le même objet, 1 sinon. L'équation 3.22 montre une fonction de coût possible associé au réseau siamois, avec $F(x)$ la sortie d'une branche du réseau pour l'entrée x .

$$L(I_1, I_2, \hat{x}) = (1 - \hat{x}) * \|f(I_1) - F(I_2)\|_2^2 + \hat{x} * \max(0, m - \|f(I_1) - F(I_2)\|_2^2) \quad (3.22)$$

Le résultat de cette équation est différent de 0 dans les cas suivants :

- $\hat{x} = 0$ et la distance $\|f(I_1) - F(I_2)\|_2^2$ est supérieure à 0.
- $\hat{x} = 1$ et la distance $\|f(I_1) - F(I_2)\|_2^2$ est inférieure à une marge m fixée.

Il faut alors, dans ces cas, mettre à jour le réseau par rétropropagation. Autrement dit, on entraîne le réseau à produire une projection f tel que les images similaires soient projetées sur le même point, et que les images différentes aient une projection séparée d'une marge m . Ainsi, pour la recherche d'instances, nous pouvons parcourir l'ensemble des images de la base de donnée et trouver les images avec une projection identique pour déterminer l'objet présent dans l'image requête.

FaceNet [15] propose d'utiliser les réseaux siamois pour comparer les visages dans les images. En utilisant cette fois-ci un réseau à trois branches, les entrées sont un ensemble de triplets plutôt que de couples. Les triplets sont composés d'une image de base (appelée *anchor*), d'une image représentant la même instance (appelée *positive*) et une image différente (appelée *negative*). L'objectif est alors de produire une projection par le réseau, telle que la représentation de l'image *anchor* soit plus proche de l'image *positive* que de l'image *negative*. On peut alors définir une nouvelle fonction de coût, avec les images I_a , I_p et I_n respectivement anchor, positive et négative. L'équation 3.23 va alors être positive lorsque $\|f(I_a) - F(I_p)\|_2^2 > \|f(I_a) - F(I_n)\|_2^2$, et elle vaudra 0 lorsque $f(I_a)$ sera plus proche de $f(I_p)$ que de $f(I_n)$, comme montré sur le schéma 3.25, pris depuis [15].

$$L(I_a, I_p, I_n) = \max(0, \|f(I_a) - F(I_p)\|_2^2 - \|f(I_a) - F(I_n)\|_2^2) \quad (3.23)$$

La figure 3.26 présente l'apprentissage d'un réseau siamois à trois branches. La première image en haut à gauche est l'image de base, la seconde l'exemple positif et la troisième l'exemple négatif. Le réseau de neurones (CNN) est présenté trois fois sur le schéma, mais il s'agit bien du même réseau. Les poids sont partagés entre les branches, et l'apprentissage est fait simultanément, le réseau est donc bien toujours le même sur les trois branches.

Schroff et al. [15] mettent également en avant le problème de sélection des triplets, qui

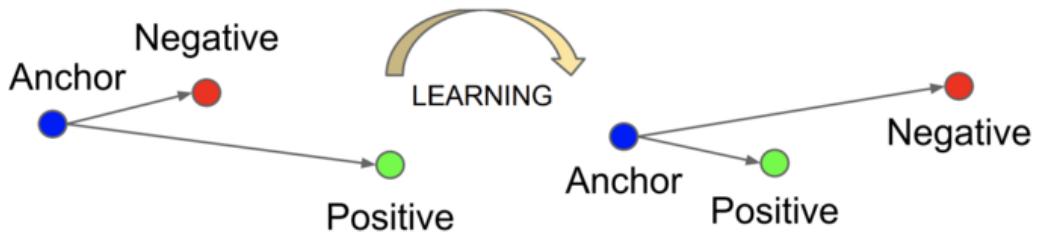


Figure 3.25 : Apprentissage des projection avec un réseau triple (source [15])

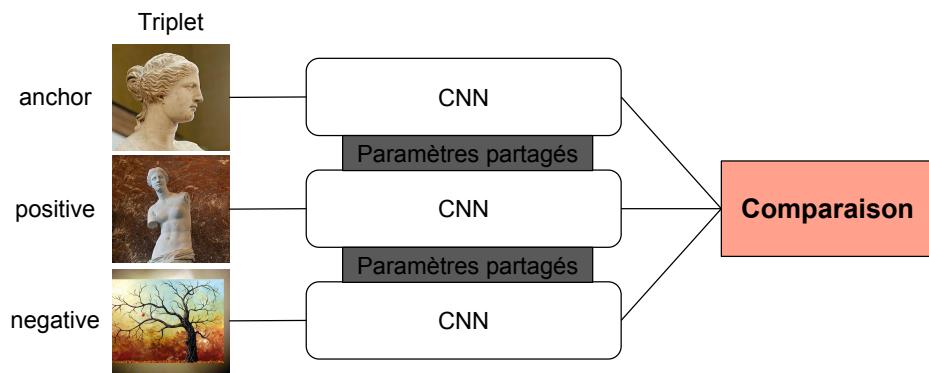


Figure 3.26 : Réseau siamois à trois branches

a un rôle très important dans la capacité à entraîner le réseau. Cette problématique est très dépendante des corpus sur lesquels est entraîné le réseau et dans le chapitre 4 nous abordons ce problème dans le cadre de nos collections.

Gordo et al. [86] se basent sur l'approche de Schroff et al. et sur celle de Tolias et al. [117], pour enrichir cette architecture avec une proposition de régions d'intérêts. En utilisant la projection d'une région d'intérêt plutôt que de l'image entière, ils sont capables de produire une projection qui capture mieux la similarité entre les objets. Nous nous basons notamment sur leur approche pour proposer notre propre architecture dans le chapitre 5.

Les réseaux siamois à trois branches sont particulièrement bien adaptés à la recherche d'instances. Ils obtiennent en effet les meilleurs résultats sur les collections courante de recherche d'instance [86]. Dans les chapitres 4 et 5, nous les explorons plus en détail et nous étudions comment les adapter à notre problème, tout en proposant une nouvelle fonction de coût et un nouveau pipeline d'apprentissage.

3.6 DÉTECTION D'ÉVÉNEMENT DANS LES VIDÉOS

Dans le cadre du projet GUIMUTEIC, en plus de la partie reconnaissance d'instance, nous nous intéressons à la reconnaissance de gestes dans les vidéos. Nous avons mis en évidence 5 gestes (6 si l'on ajoute la détection de l'absence de geste), qu'il faut identifier (section 2.2.1). L'approche standard de la classification vidéo comporte trois étapes principales [100, 124, 125], qui est très proche de celle présentée précédemment sur la classification d'image :

- Extraction de caractéristiques visuelles locales avec des descripteurs visuel, qui en plus de ceux présentés précédemment peuvent être temporelles [126, 127].
- Création d'une description pour chaque segment de vidéo, par exemple à l'aide de sac de mots visuels [128].
- Entraînement d'un classifieur (SVM généralement) pour la reconnaissance d'actions.

Ce pipeline a les mêmes désavantages que ceux présenté dans la section 3.5.1, et le fait qu'ils se basent sur des descripteurs ingénierés est l'une des limitations principales. Les premières application de CNN pour la détection d'action dans les vidéos consiste à considérer l'entrée comme une image avec une dimension de plus (le temps), et d'appliquer des convolutions 3D à la place des convolutions 2D présentées précédemment. Une convolution 3D est similaire à une convolution 2D, et l'architecture utilisée est alors similaire à celle d'un réseau profond type AlexNet [129]. Les caractéristique visuelles spatiales et temporelles sont donc considérées de la même manière. D'autres approches proposent de traiter différemment les données spatiales et temporelles avec deux réseaux distincts [130], avec un réseau qui s'intéresse uniquement à l'image et l'autre qui regarde le flux optique de la vidéo. Le principal problème ici vient du fait qu'il faut calculer le flux optique pour chaque image de la vidéo, et dans un contexte mobile cela se révèle très coûteux, et rédhibitoire pour une application en temps réelle.

Karpathy et al. [16] proposent de traiter les données temporelles à l'intérieur du réseau de neurones. La figure 3.27 présente différentes manières de réaliser la fusion des information des informations temporelles, avec en bas la série temporelle d'image composant la vidéo et en haut la classification issue du réseau. Le *single frame* correspond à faire de la classification d'image classique comme présenté précédemment. Ne prenant en compte aucune donnée temporelle, il s'agit de celui qui donne les moins bons résultats, et sert généralement de baseline. La *late fusion* prend deux trames, les fait passer chacune dans le réseau et, grâce aux couches entièrement connectées finale, faire une fusion. Le désavantage principal de cette technique est l'augmentation du temps de calcul, avec chaque trame

lu devant être passer dans le réseau. De plus, tout le travail de fusion est réalisé par une couche entièrement connectée, ce qui fait beaucoup de paramètres supplémentaire, avec une solution optimale potentiellement plus difficile à trouver. La *early fusion* consiste à considérer l'entrée non plus comme une image, mais comme une matrice de dimension 3, ce qui revient à concaténer les images images à la suite les unes des autres en entrée. L'avantage principal est le nombre de paramètres du réseau qui n'augmente que très peu, mais le problème vient de la perte rapide d'information temporelle. Enfin la *slow fusion* présente les meilleurs résultats, et propose de fusionner les informations temporelles au fur et à mesure dans le réseau. Cette technique est une intégration des deux précédente, en prenant leurs avantages respectifs, sans les inconvénients.

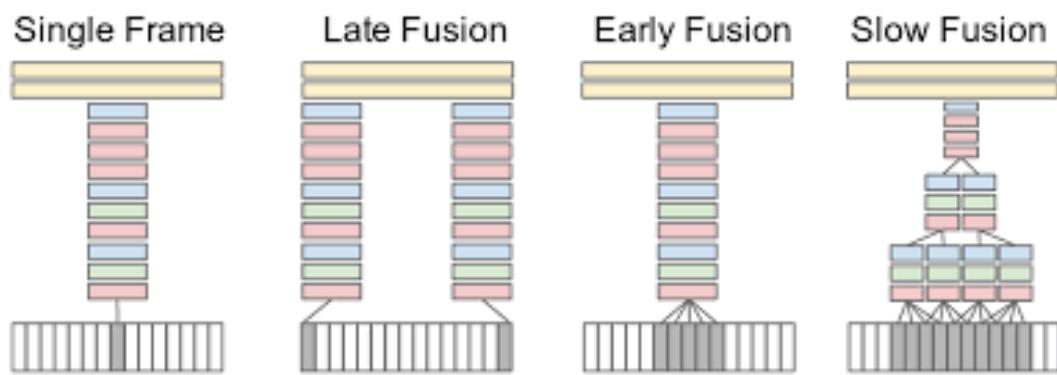


Figure 3.27 : Différentes méthodes de fusion d'information temporelle avec un CNN (source [16]).

Ce type d'approches cependant est généralement dépassé en terme de performance par des réseaux récurrents (RNN) [18]. Ce type de réseau connecte chaque entrée aux entrées précédentes, ce qui lui permet de garder une “mémoire” des informations passées. Sur la figure 3.28, on voit une représentation simplifiée d'un tel réseau récurrent. Chaque entrée, une image dans notre cas, est passée dans le réseau, en même temps qu'une sortie du réseau sur une des entrées précédentes, que l'on nomme état caché.

Une limitation significative des modèles RNN simples est la disparition du gradient (*vanishing gradient* [131]). La capacité à rétro-propager un signal d'erreur à travers le temps devient de plus en plus difficile. De la même manière qu'avec les réseaux très profonds type ResNet, le gradient devient trop faible à mesure que le réseau devient profond. Les blocs LSTM (Long Short-Term Memory) permettent un apprentissage plus profond grâce à des mécanismes non linéaire pour transmettre l'état caché, qui est alors soit propagé sans modification, être mise à jour ou réinitialisé grâce à des portes de connection [132].

Les LSTM, et leur dérivés comme les GRU [133], ont permis des avancées dans la recon-

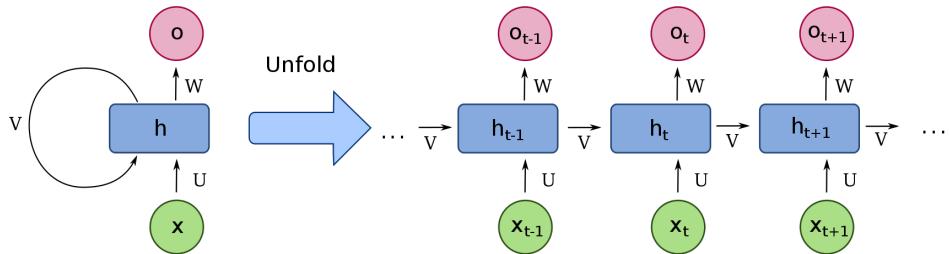


Figure 3.28 : Schéma d'un réseau récurrent. A droite la version compression, à gauche la version dépliée sur chaque entrée (source : [17])

naissance d'action et d'évènement dans les vidéos. Couplés avec un CNN pour réaliser l'extraction de caractéristique spatiale, les LSTM font office de fusion d'information temporelle. On voit sur la figure 3.29 l'utilisation typique d'un CNN et d'un LSTM pour classifier un segment de vidéo. Bien que ces approches présentent généralement les meilleures performances [18, 134–136], elles présentent le problème d'être coûteuse en terme de calcul, et de plus, nécessite que chaque trame de la vidéo soit passée dans le réseau de neurones, ce qui augmente grandement le nombre de calcul à effectuer. Dans le cadre d'une application mobile, pour limiter le nombre de paramètres, l'occupation mémoire et le temps de calcul, nous nous basons dans le chapitre 6, sur des méthodes à bases de CNN uniquement.

3.7 CONCLUSION

Nous avons présenté dans ce chapitre les méthodes existantes pour la recherche d'instances. Dans les chapitres 4 et 5, nous présentons nos propositions pour adapter ces méthodes à nos problématiques, et améliorer les résultats dans le cas de la recherche d'instances en mobilité.

La proposition de Gordo et al. [86], présente les meilleurs résultats pour la recherche d'instances. Cependant elle requiert l'utilisation de régions annotées sur les images, ce dont nous ne disposons pas. Dans le chapitre suivant, nous adaptons les réseaux siamois à trois branche, avec un modèle proche de celui de FaceNet, avec une sélection de triplets adaptée à nos collection. Les réseaux à proposition de régions présentent les meilleurs résultats et donc, dans le chapitre 5, nous étudions les possibilité de régions sans annotation dans la collection.

Pour la détection de gestes, dans le chapitre 6, nous n'utilisons pas les réseaux récursifs, par un soucis de temps de calcul et d'occupation mémoire sur processeur mobile. Nous

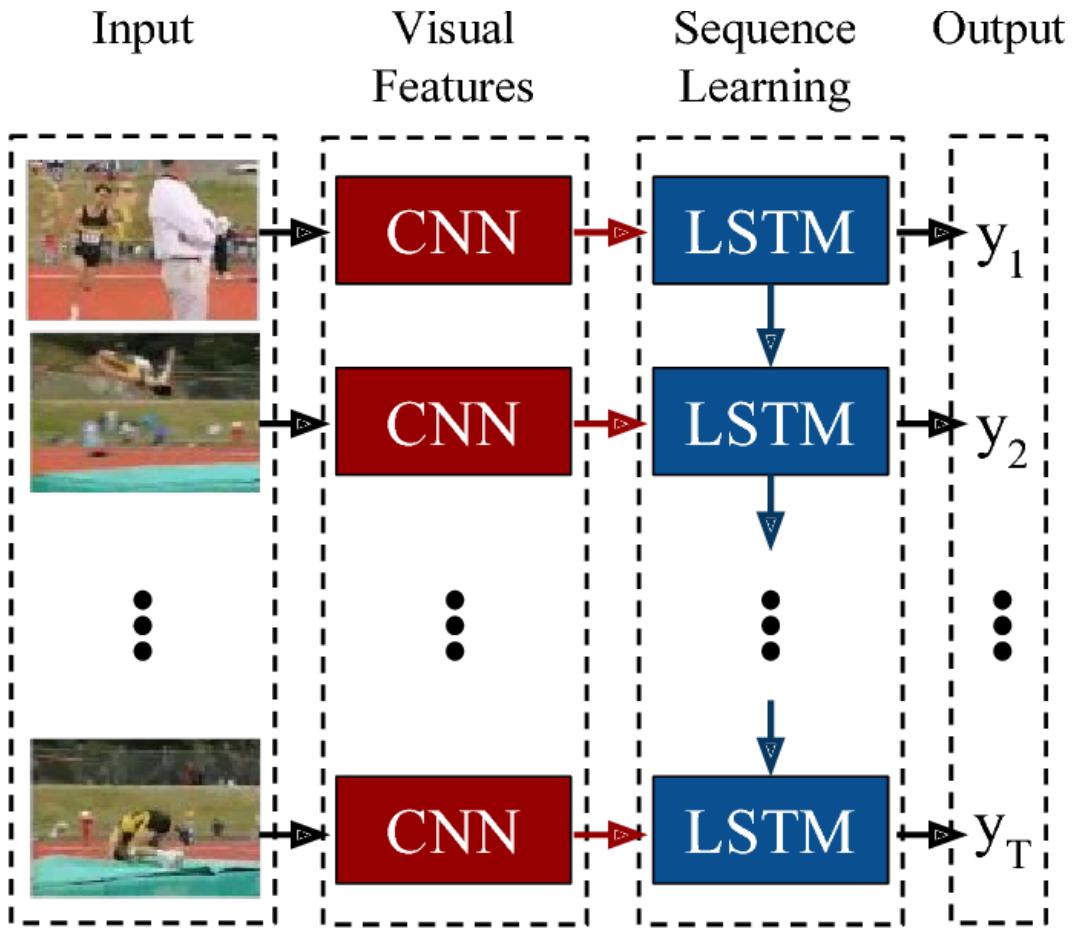


Figure 3.29 : Réseaux récurrents pour la détection d'action dans les vidéos à base de CNN et de LSTM (source [18])

nous intéressons à la fusion d'information temporelle à l'intérieur du réseau à la manière de Karpathy et al. [16], avec comme objectif de réduire le nombre de paramètres.

4

Identification d'œuvres

Comme énoncé précédemment, le système GUIMUTEIC doit être capable de reconnaître l'environnement de l'utilisateur afin de lui donner les informations qu'il désire sur ce qui l'entoure. Pour identifier cet environnement, nous nous basons sur la reconnaissance d'œuvres, ou instances, et de points d'intérêts. Cette reconnaissance d'instances, présentée dans la chapitre 3, peut être faite par recherche d'images. Ce chapitre étudie le problème de reconnaissance d'instances dans les images. Notre approche, basée sur l'apprentissage de similarité entre les images, repose sur les réseaux de neurones profonds de type siamois à trois branches.

4.1 SIMILARITÉ ENTRE LES IMAGES

Le reconnaissance d'instances consiste à identifier, sur une image donnée, le ou les objets présents. Elle se distingue de la classification d'images, dans le sens où on ne s'intéresse pas à une catégorie d'objet, mais à un objet bien identifié. Longtemps basé sur les comparaisons d'images à l'aide d'extraction de caractéristiques visuels sur celles-ci, l'utilisation d'apprentissage automatique donne aujourd'hui les meilleurs résultats, comme montré dans le chapitre 3.

La reconnaissance d'instances peut se faire à l'aide de recherche d'images, où l'on cherche

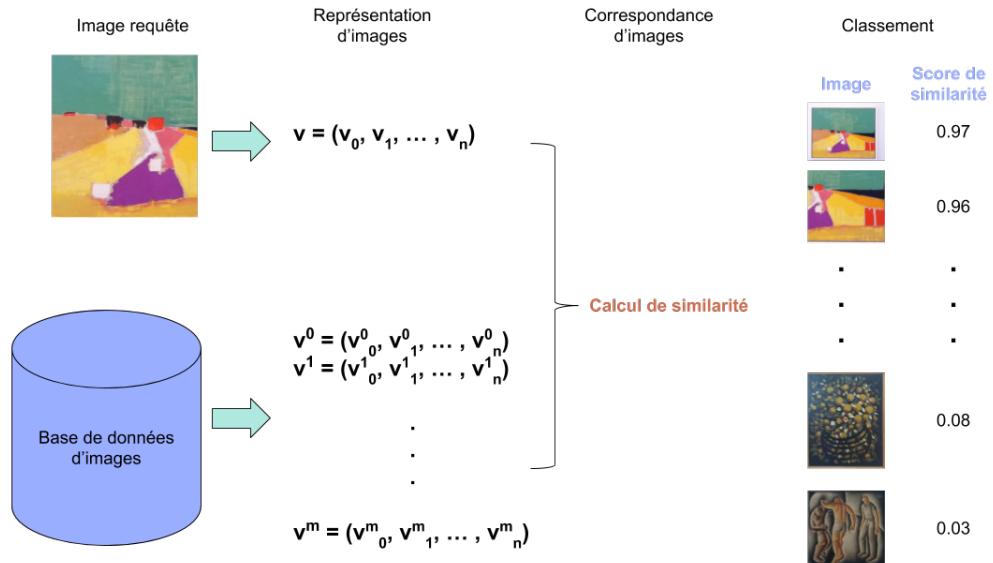


Figure 4.1 : Exemple fictif de recherche d'image par le contenu. Le classement représente la similarité entre les images.

dans une base de données les images les plus “ressemblantes”, pour déterminer quel objet est visible sur l’image. Cette tâche revient donc à déterminer ce qu’est la ressemblance entre deux images. Dans notre cas, similarité signifie avoir la même instance. Dans le cas idéal, nous souhaiterions avoir la fonction similarité S définie par :

$$S(I_1, I_2) = \begin{cases} 1 & \text{si } I_1 \text{ et } I_2 \text{ contiennent le même objet} \\ 0 & \text{sinon} \end{cases} \quad (4.1)$$

Dans la suite de ce chapitre, le but sera d’approximer cette fonction S . Une fonction approximée de S tend vers 1 si les images sont similaires. Pour déterminer l’objet présent dans les images, nous retrouvons les images similaires dans la base de données. Comme montré sur le schéma 4.1, la recherche d’image permet de classer les images, et ainsi définir la fonction de similarité en fonction de score de ressemblance entre les images.

Nous voyons sur le schéma 4.1 que pour calculer la similarité des images, nous avons besoin de créer une représentation commune des images V . Nous avons exploré dans le chapitre 3 différents moyens de créer ses représentations, à l’aide de descripteurs visuels, ou par apprentissage automatique. En ce basant sur cet état de l’art, il apparaît que les méthodes à base de réseaux de neurones profonds, notamment grâce aux réseaux siamois, permettent les meilleurs résultats.

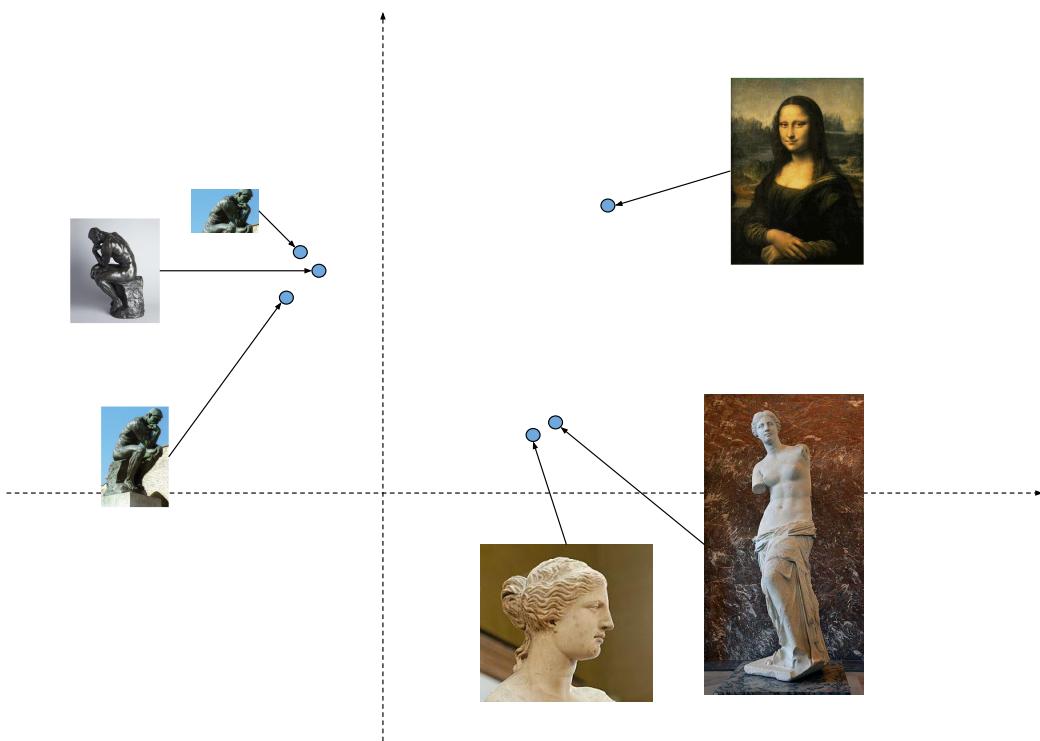


Figure 4.2 : Projection d'images dans un plan 2D

4.2 REPRÉSENTATION D'IMAGE ET SIMILARITÉ

Pour comparer les images, celles-ci doivent avoir une représentation commune qui permet un calcul de similarité. Pour créer une représentation commune des images, nous créons un espace de projection de dimension N , où chaque image est représentée par un vecteur V , son plongement. La caractéristique principale de cet espace est que les distances entre les vecteurs doivent être inversement proportionnelles à la similarité entre les images. La distance entre V_1 et V_2 doit être minimale si l'image 1 et l'image 2 représentent le même objet. Comme montré sur le schéma 4.2, le plongement dans cet espace doit regrouper les images similaires ensemble, et les éloignées des autres. Ainsi, en projetant l'image requête dans cette espace, nous pouvons déterminer l'objet présent dans celle-ci, à l'aide de la recherche du plus proche voisin.

La dimension N de l'espace de projection a une forte importance dans notre contexte. Cette dimension va déterminer une partie du temps de recherche des images. Plus cette dimension est grande, plus les temps de calcul seront importants. De la même manière, le type de métriques utilisé pour le calcul de similarité va impacter le coût de calcul. L'avantage d'utiliser une méthode d'apprentissage automatique, est que nous pouvons établir la fonction objectif de l'apprentissage en fonction de la métrique qui nous intéresse.

4.3 PLONGEMENT DES IMAGES

Nous souhaitons apprendre une projection des images, un plongement, qui capture la similarité entre les images. Pour le construire, nous utilisons un réseau de neurones, dont la sortie est la projection de dimension N . Nous avons montré dans le schéma 3.21 de la section 3.5.2 qu'il est possible d'extraire les caractéristiques depuis n'importe quelle couche cachée d'un réseau de neurones pour créer un représentant de l'image. Dans notre cas, nous voulons créer un plongement $V \in \mathbb{E}$ qui projette l'image dans un espace euclidien \mathbb{E} de dimension N .

Nous créons un réseaux de neurones qui produit le plongement P des images. Pour l'apprentissage de ce réseau, nous nous basons sur la seule information dont nous disposons, c'est-à-dire un ensemble d'images avec leur étiquette pour savoir si elles représentent le même objet. Etant donnée une image I parmi l'ensemble des images disponibles τ , nous définissons l'ensemble des images $\tau_p(I)$ contenant le même objet que l'image I (équation 4.2), et inversement $\tau_n(I)$ l'ensemble des images ne contenant pas le même objet que I (équation 4.3).

$$\tau_p(I) = \{x \in \tau | S(I, x) = 1\} \quad (4.2)$$

$$\tau_n(I) = \{x \in \tau | S(I, x) = 0\} \quad (4.3)$$

4.4 RÉSEAU SIAMOIS

Dans le chapitre 3, nous avons détaillé l'apprentissage à base de réseaux siamois. Ces derniers permettent d'apprendre une similarité entre les images [15]. Nous nous basons sur les réseaux siamois à trois branches, présentés sur le schéma 3.26. Ceux-ci obtiennent les meilleures performances pour la recherche d'instance [86], et sont plus adaptés pour les collections de taille réduite.

Là où Schroff et al. [15] peuvent se baser sur une grande collection d'image de visage pour apprendre les différences, nous ne disposons pas de corpus aussi important pour la recherche muséale. Pour être capable d'utiliser des réseaux profonds de grandes tailles, qui fournissent généralement les meilleures résultats, nous devons utiliser du transfert de connaissances (présenté en section 3.5.3). Comme nous allons réaliser une première partie de l'apprentissage sur ImageNet [56], nous basons donc notre architecture sur des réseaux convolutionnels qui sont capables d'obtenir de bonnes performances sur cette collection. Une fois l'apprentissage réalisé sur ce corpus, nous pouvons spécialiser notre réseaux pour

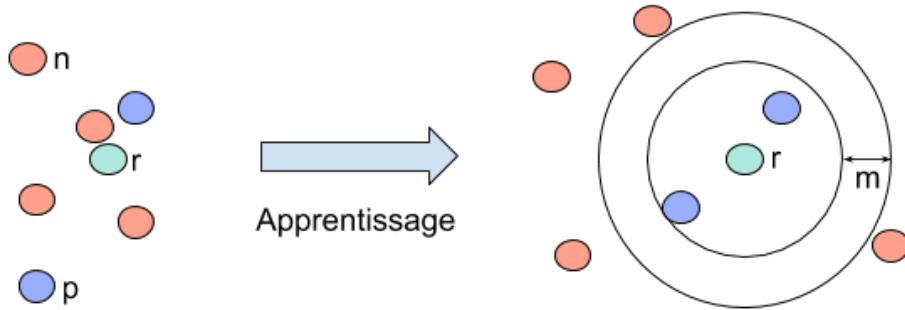


Figure 4.3 : Schéma de l'apprentissage du plongement des images, pour une image de références r , avec les images positives p représentant le même objet, et les images n qui contiennent des objets différents.

créer le plongements P qui nous intéressent. Cette phase d'apprentissage se fait à l'aide de triplets d'images. Ces triplets représentent (équation 4.4) une image de **référence** I_r , une image **positive** I_p représentant le même objet que l'image référence, et une image **négative** I_n qui contient un objet différent.

$$H = \{(x, y, z) \in \tau^3 | y \in \tau_p(x) \text{ et } z \in \tau_n(x)\} \quad (4.4)$$

Le réseau de neurones produit une fonction P de plongement des images dans \mathbb{E} . Pour entraîner ce réseau, nous devons définir une **fonction objectif** associée. Cette fonction doit capturer la similarité entre les images. Nous voulons que $P(I_r)$ soit plus proche du plongement $P(I_p)$ que de $P(I_n)$, comme montré par l'équation 4.5, avec m qui représente la marge minimale désirée entre les plongements positifs et négatifs :

$$\forall (x, y, z) \in H, \|x - y\| + m < \|x - z\| \quad (4.5)$$

Cette marge m nous permet de définir un seuil, qui nous permet de déterminer la similarité S :

$$S(I_1, I_2) = \begin{cases} 1 & \text{si } \|P(I_1) - P(I_2)\| < m \\ 0 & \text{sinon} \end{cases} \quad (4.6)$$

Nous nous intéressons uniquement à la projection sur l'hypersphère unité des plongements (équation 4.7). Ce qui permet d'utiliser le produit scalaire uniquement pour calculer

la similarité cosinus entre les embeddings et mène à un calcul de gradient moins coûteux.

$$\forall I \in \tau, P(I) = \frac{P(I)}{\|P(I)\|}, \text{ donc } \|P(I)\| = 1 \quad (4.7)$$

Pour l'apprentissage, nous souhaitons donc à minimiser l'équation **objectif triple** :

$$\mathcal{L}(x, y, z) = \max(0, P(x) \cdot P(z) - P(x) \cdot P(y) + m) \quad (4.8)$$

pour l'ensemble des triplets $(x, y, z) \in H$. Cette fonction de coût est grande si z est similaire à x , autrement dit elle est proportionnelle au produit scalaire $P(x) \cdot P(z)$. Et elle diminue si y est similaire à x , c'est-à-dire qu'elle est inversement proportionnelle à $P(x) \cdot P(y)$. Ce qui correspond sur le schéma 4.3, à éloigner les exemples négatifs et rapprocher les exemples positifs pendant l'entraînement. La fonction P produite par le réseau de neurones doit donc faire tendre la fonction objectif triple vers 0.

4.5 CHOIX DES TRIPLETS

Lorsque l'on utilise une fonction objectif triple il faut être particulièrement attentif à la sélection des triplets [15]. La qualité de l'apprentissage et la vitesse de convergence vont dépendre des triplets présentés au réseau pendant l'apprentissage. Plus précisément, il existe des exemples plus ou moins "faciles". Il nous faut choisir les triplets pour l'apprentissage pour deux principales raisons : un triplet facile ne fait rien apprendre au réseau et des triplets trop difficiles dès le départ conduisent rapidement à un minimum local ou font s'effondrer le réseau ($P(x) = 0, \forall x$).

Un exemple facile, à gauche sur le schéma 4.4a correspond à un triplets où les projections sont correctes. Au centre, un exemple difficile correspond au cas où le plongement de l'image négative est plus proche de l'image de référence que celui de l'image positive, et comprend également tous les cas où l'ordre entre positif et négatif n'est pas respecté. Enfin, nous référerons à un exemple semi-difficile, à droite, dans le cas où les distances entre les embeddings est correcte, mais l'exemple négatif est dans la marge m .

En nous basant sur l'équation 4.5, nous pouvons définir un exemple positif facile ou difficile en fonction de la valeur de $\|P(x) - P(y)\|$ pour un couple d'image (x, y) représentant le même objet. Les exemples positifs difficiles p_d sont ceux dont la projection est plus élo-

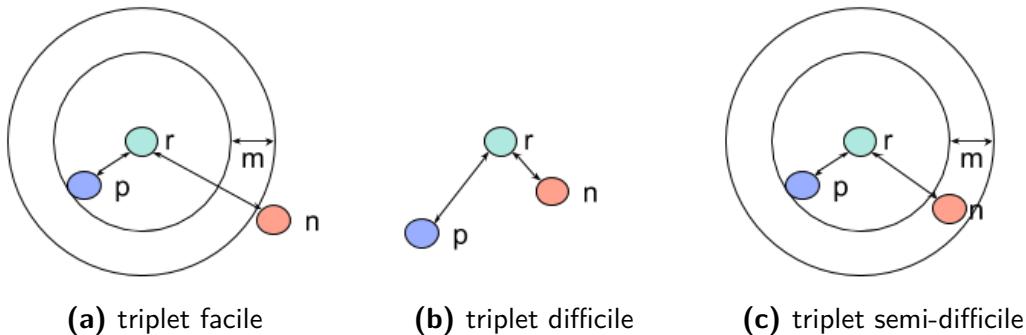


Figure 4.4 : Exemple des différentes configuration des projections possibles. La projection de l'image de référence r est plus ou moins proche des projection de l'image positive p ou de l'image négative n .

gnée qu'au moins un des exemples négatifs :

$$p_d(x) = \{y \in \tau_p(x) \mid \exists z \in \tau_n(x), \|P(x) - P(y)\| > \|P(x) - P(z)\|\} \quad (4.9)$$

Nous définissons de la même manière les exemples négatifs difficiles n_d en fonction de la valeur de $\|P(x) - P(z)\|$, où les images x et z représentent des objets différents :

$$n_d(x) = \{z \in \tau_n(x) \mid \exists y \in \tau_p(x), \|P(x) - P(y)\| > \|P(x) - P(z)\|\} \quad (4.10)$$

L'ensemble les **triplets difficiles** H_d (équation 4.11) correspond aux triplets pour les-quels les exemples positifs et négatifs sont diffiles.

$$H_d = \{(x, y, z) \in H \mid y \in p_d(x) \text{ et } z \in n_d(x)\} \quad (4.11)$$

Une première stratégie pour la selection des triplets, proposée par Schroff et al. [15], consiste à prendre les triplets semi-difficiles, en choisissant les positifs les plus difficiles, et de prendre les exemple négatifs qui sont dans la marges m . Une autre méthode par Gordo et al. [86], propose de choisir les i exemples positifs les plus faciles et les j négatifs les plus difficiles, de choisir parmis toutes les combinaisons possibles de triplets celui qui maximise la fonction objectif triple. Ce choix semble très pertinent, car il permet de déterminer les exemples qui feront converger le plus rapidement, tout en élimant les images positifs difficiles, c'est-à-dire celles qui représentent la même instance, mais avec trop de différences visuelles. Cette méthode demande cependant de parcourir l'ensemble des données d'apprentissage pour calculer la fonction de coût. Comme le réseau est mis à jour à chaque

rétro-propagation, il faut recalculer la fonction de coût assez fréquemment.

Avec nos contraintes spécifiques, à savoir surtout que nous ne disposons que d'un ensemble limité d'exemples positifs, nous choisissons une méthode de sélection des triplets hybride entre les deux présentée précédemment. Nous choisissons l'ensemble des triplets H_{sd} , où pour l'ensemble des couples positifs possibles, nous choisissons les exemples négatifs semi-difficiles :

$$H_{sd} = \{(x, y, z) \in H \mid \|P(x) - P(y)\| < \|P(x) - P(z)\| < \|P(x) - P(y)\| + m\} \quad (4.12)$$

La figure 4.5 montre les étapes de l'apprentissage du réseau. Dans un premier temps, le réseau est appris sur une base de données de grande taille, par exemple ImageNet. La deuxième étape correspond en un fine-tuning sur la collection du musée. Ceci permet de spécifier le réseau aux images de musée, étant donné la différence entre les deux collections, ce qui rend la convergence à l'étape suivante plus rapide. Finalement, nous apprenons le réseau à l'aide des triplets d'image du musée.

Dans un second temps, une fois qu'il n'y a plus de convergence, nous choisissons tous les triplets de H_d .

4.6 ÉVALUATION

Pour évaluer notre première proposition, nous utilisons la collection CLICIDE (détalée dans la section A.2). Ce corpus représente un ensemble de photos d'un musée d'art, et donc surtout des tableaux et peintures, avec différents points de vue sur les œuvres, et correspond aux contraintes liées à notre projet. La collection GaRoFou (section A.2) vient du musée d'archéologie de Lyon Fourvière, et présente des objets de différentes natures, comme des sculptures, des artéfacts ou des stèles.

Nous comparons notre approche à celles de l'état de l'art, et à diverses méthodes de deep learning. Les méthodes à base de descripteurs visuels sur ce dataset ont été explorées précédemment [28], avec l'utilisation de SIFT et de sacs de mots visuels. Pour identifier l'image similaire dans la base d'image qui va nous permettre de déterminer l'objet présent dans l'image, on recherche l'image la plus proche, ce qui correspond au TOP@1 de la recherche d'image :

$$TOP@1(I) = \arg \min_{x \in \tau} \|P(I) - P(x)\| \quad (4.13)$$

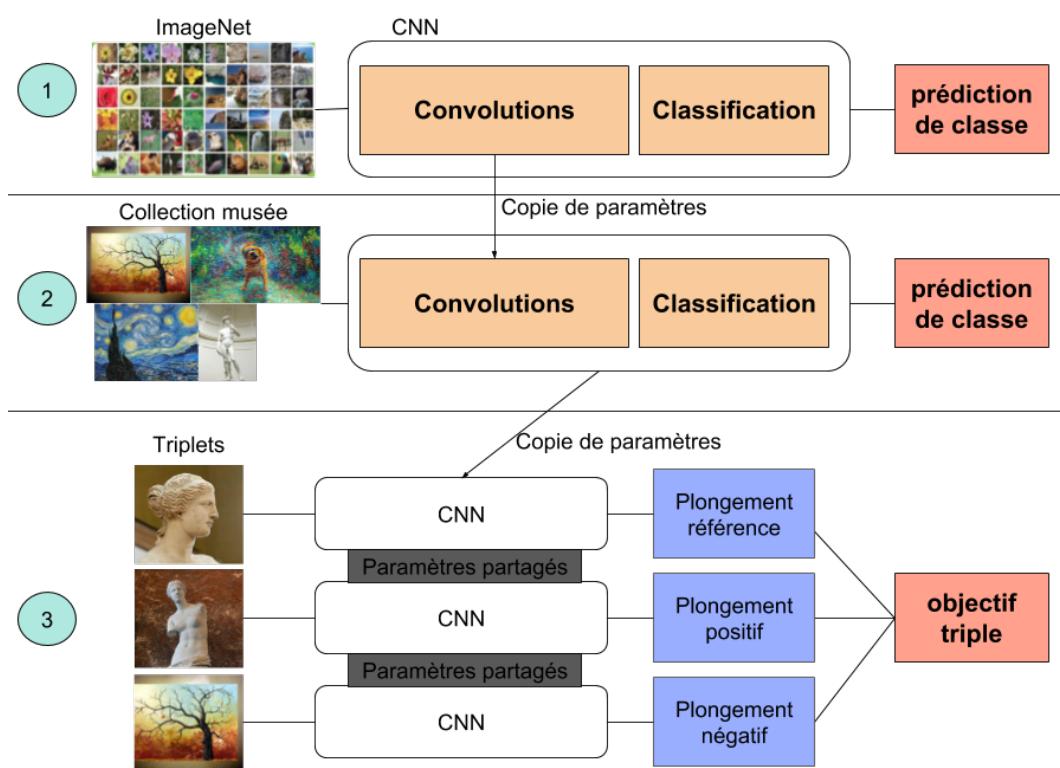


Figure 4.5 : Pipeline d'apprentissage. L'étape 1 correspond à l'apprentissage du réseau sur une grande collection d'image, ici ImageNet. L'étape 2 est le fine-tuning sur la collection du musée. La troisième étape consiste à apprendre à l'aide du réseau siamois triple.

4.6.1 AUTRES APPROCHES COMPARÉES

Toutes les méthodes comparées utilisent le même type d'architectures, à savoir AlexNet et Resnet, présentés plus en détail dans le chapitre 3. Nous nous comparons principalement aux réseaux siamois à trois branches proposés par Gordo et al. [86], avec multi-resolution(Gordo multi-res) et sans (Gordo). Pour avoir une idée des autres techniques utilisables, nous montrons les résultats obtenus avec :

- Extraction de caractéristiques : Alexnet E et Resnet E (pour Extraction). Un réseau de neurones de type Alexnet ou Resnet est appris sur une grande collection, ici ImageNet. Les images du musée sont ensuite représentées par la sortie d'une des couches du réseau. Nous sélectionnons ici la couche cachée qui donne les meilleurs résultats dans notre cas, à savoir la couche *fc6* pour Alexnet et *pool5* pour Resnet.
- Fine-tuning et classification des œuvres : Alexnet Classif et Resnet Classif. Un réseau est appris de manière conventionnelle sur ImageNet, et ensuite un apprentissage fin est réalisé où chaque œuvre est une classe. Les résultats sont ceux de la classification de chaque œuvres du corpus de test.
- Réseau siamois à double branche : Alexnet SS et Resnet SS (pour Siamois Simple). Un réseau siamois à deux branches est utilisé pour l'apprentissage. La sélection des couples se fait de la manière suivante : tous les couples positifs possibles, et le même nombre de couple négatifs, qui correspondent aux exemples les plus difficiles.

4.6.2 PARAMÈTRES D'APPRENTISSAGES

Le fine-tuning sur les deux réseaux utilisés se fait en commençant par la dernière couche entièrement connectée. Nous apprenons tout d'abord la dernière couche seule, et après stabilisation de la fonction de coût, nous ajoutons à l'entraînement la couche précédente. Nous continuons ainsi jusqu'à avoir entraîné toutes les couches. Il est important de ne pas entraîner tous les réseaux d'un seul coup, car lorsque l'on fait un *fine-tuning*, on supprime la dernière couche de classification pour la remplacer par une qui a le bon nombre de sorties. Cette nouvelle couche étant initialisée aléatoirement, l'erreur de classification sera grande, et propager cette erreur risque de détruire l'ensemble du réseau. Il est donc nécessaire de procéder couche par couche, car les premières couches sont moins susceptibles de devoir être changée, étant donné qu'elles capturent des caractéristiques de plus bas niveau, plus proche de l'image. De plus, entraîner tout le réseau sur une petite collection entraîne un sur-apprentissage, les premières couches ne devraient donc pas être modifiée.

Ces considérations mènent à l'apprentissage suivant :

- Pour AlexNet : apprentissage depuis la dernière couche jusqu'à la dernière couche

convolutionnelle. Les premières couches convolutifs ne sont pas modifiées.

- Pour ResNet : apprentissage de deux derniers blocs de convolution. Contrairement à AlexNet, ResNet repose sur moins de couches entièrement connectées, nous devons donc entraîner davantage de couches convolutionnelles. Comme cette architecture est basée sur des blocs de convolution, nous nous référerons à un bloc en entier.

4.6.3 ETUDE DE L'IMPORTANCE DE L'AUGMENTATION DE DONNÉES D'APPRENTISSAGE

Toutes les approches présentées se basent sur un fine-tuning. L'apprentissage sur le corpus de départ, ainsi que sur notre collection, est fait avec une partie d'augmentation de données. Cette augmentation consiste en différentes perturbations faites sur les images pour éviter un sur-apprentissage du réseau. Autrement dit, si notre réseau sur-apprend, il aura une capacité limitée à s'adapter à des données différentes de celle d'apprentissage. Comme nous nous basons sur du transfert de connaissance pour être capable de travailler sur nos collections de petites tailles, cette généralisation est un élément important de notre pipeline.

L'augmentation de données a deux objectifs. Le premier est d'augmenter le nombre de données présentées au réseau de neurones. La performance des réseaux de neurones est très fortement liée à la quantité de données sur laquelle on travaille. Particulièrement dans notre cas, cette augmentation nous permet d'éviter un sur-apprentissage très prononcés sur les petites collections. Le deuxième objectif est de pallier au manque de variation dans les images de la base d'entraînement. En simulant des déplacements de caméra, d'effet de zone, de rotation, on aide le réseau à être invariant à ces transformations. Trois différentes transformations d'images sont aléatoirement appliquées pendant l'entraînement :

- Rotation des images avec une probabilité équivalente pour tous les angles $\{-180, -90, 0, 90, 180\}$.
- Changement d'échelle des images avec un facteur aléatoire pour chaque dimension dans l'intervalle $[0.75, 1.25]$.
- Renversement d'image horizontal avec probabilité de 0.5

Le tableau 4.1 présente l'influence de chacune de ces perturbations, avec le réseau ResNet-152.

Le renversement d'image est particulièrement intéressant dans notre cas. Il est souvent utilisé dans l'apprentissage de classe, où l'invariance à l'orientation semble important, dans le cas de reconnaissance d'instance cela semble contre-intuitif. Notamment dans le cadre de collection de tableaux ou de peintures, nous ne voulons pas confondre deux objets qui pourraient être l'image inversée l'un de l'autre. Par exemple, l'œuvre "4900 Farben" ("4900

<i>Rotation</i>	<i>Scaling</i>	<i>Flipping</i>	<i>Mean Precision@1 (in %)</i>	
			<i>CLICIDE</i>	<i>GaRoFou</i>
			72.12	92.93
✓			74.55	94.02
	✓		76.36	93.48
		✓	72.12	94.57
✓	✓		76.97	94.57
✓		✓	75.75	94.57
	✓	✓	78.18	93.48
✓	✓	✓	79.39	94.57

Table 4.1 : Influence de la rotation, de la mise à l'échelle et du renversement sur les résultats avec l'architecture ResNet-152.

Colors”)¹ de Gerhard Richter est constituée d'un ensemble de panneaux de carrés de couleurs. Dans ce cas, le renversement d'une image n'est pas équivalente à l'image d'origine, et ne devrait pas être considéré dans l'apprentissage.

Le tableau 4.1 montre que la rotation seule ou la mise à l'échelle seule améliorent les résultats. Cependant, comme attendu, le renversement seul n'améliore pas l'apprentissage. Mais n'importe quelle combinaison de perturbation donne de meilleurs résultat que toute les transformations seules, et ce même en considérant le renversement. De ces résultats nous pouvons conclure que n'importe quelle augmentation de données à travers des perturbations est toujours utile, même pour la reconnaissance de peinture dans le cadre de la collection CLICIDE. Pour la collection GaRoFou, n'importe quelle perturbation aide l'apprentissage, mais la combinaison de différente transformations ne change pas les résultats. Cela peut être dû à la plus grande diversité des objets présents dans la collection, ainsi qu'à leur caractère tri-dimensionnel.

4.6.4 RÉSULTATS

Tout d'abord, nous pouvons voir sur le tableau 4.2 que les méthodes à base de descripteurs visuels ingénierés sont dépassés par toutes les méthodes à base de réseaux de neurones, même si nous travaillons sur une collection d'image de taille réduite. Pour toutes les méthodes proposés, exception faite de l'extraction de caractéristique, une architecture de type Resnet obtient de meilleurs résultats. Pour l'extraction de caractéristiques depuis un réseau pré-entraîné, il en va de la nature de ces réseaux qui explique que Alexnet ait de meilleures performances. Là où Alexnet est une suite de convolution et de maxpooling, Re-

1. <https://www.gerhard-richter.com/en/art/microsites/4900-colours>

	<i>TOP@1 (in %)</i>
	CLICIDE
<i>Descripteurs Visuels [28]</i>	70.08
<i>Gordo [86]</i>	90.3
<i>Gordo multi-res [86]</i>	92.73
<i>AlexNet E</i>	72.73
<i>AlexNet Classif</i>	78.18
<i>AlexNet SS</i>	75.76
<i>ResNet E</i>	72.12
<i>ResNet Classif</i>	79.39
<i>ResNet SS</i>	85.45
<i>Architecture AlexNet</i>	79.39
<i>Architecture ResNet</i>	92.73

Table 4.2 : Évaluation des différentes méthodes d'apprentissage pour la projection des images sur le collection CLICIDE.

snet est une série de bloc avec des skip-connection. Que l'extraction sur une couche donnée donne de moins bons résultats signifie juste que dans ce cas les réseaux de retiennent pas l'information de la même manière. Il est intéressant de noter que malgré la petite taille de la collection et d'un grand manque de diversité dans les images (toutes avec le même fond, dans le même musée, etc), le fine-tuning reste intéressant pour améliorer les résultats.

Nous remarquons surtout que les meilleurs résultats sont obtenu avec notre approche, mais également avec l'approche de Gordo. Bien que ces résultats soient surprenamment haut pour cette approche, car les embeddings ne sont pas appris pour cette collection en particulier, plusieurs arguments peuvent expliquer les de si bon résultats. Tout d'abord, le réseau proposer par Gordo est appris sur la collection ..., qui dispose d'un grand nombre d'exemples et d'une grande variabilité, qui se prête donc bien à la généralisation sur un autre corpus. Dans cette collection, une forte disparité entre les objets existe, ainsi que de très grand écart de point de vue et de distance de zoom. Il n'est donc pas étonnant que les plongements créés soient capables d'exprimer de manière convaincante la similarité entre les images. De plus, ce réseau dispose d'une partie de proposition de région d'intérêts. Dans le cadre de la recherche d'instance dans les images, la détection de la position de l'objet dans l'image est un élément clef, que nous n'avons pas pour le moment pris en compte dans notre approche. Lors de la création de l'embedding de l'image, nous voulons que celui ci représente l'objet présent dans l'image au mieux, en faisant fi du fond, d'autres objets moins visibles, ou du bruit.

Nous avons proposé une nouvelle fonction objectif, basé sur le produit scalaire, ainsi qu'une nouvelle sélection de triplets, le tout dans un nouveau pipeline d'apprentissage

adapté à notre problème, et à nos collections. Notre proposition obtient des résultats proches de ceux de l'état de l'art, avec un réseau plus simple, sans proposition de région, dû aux limitations d'annotation des collections. Avoir un réseau capable de détecter les régions d'intérêt, et de créer une représentation uniquement de la région où se situe l'objet semble être un éléments clé pour produire un plongement efficace. Dans le chapitre suivant, nous nous intéressons à comment mettre en place un système de détection de régions d'intérêt en plus de ce que nous avons proposé dans ce chapitre.

5

Detection de régions d'intérêts

Dans une collection d'images représentant un certains nombre d'objets, les prises de vue peuvent être multiples, de différents angles et à une distance pouvant variée. Il devient donc utile de s'intéresser à la position de l'objet dans l'image, que nous appelons région d'intérêt. Faire un plongement uniquement sur la partie de l'image contenant l'objet permet d'éliminer les autres objets pouvant être présents sur l'image, par exemple des objets à peine visible ou hors mise au point. Le plongement ainsi réalisé est plus à même de représenter correctement l'objet que le plongement de l'image entière. Dans le cadre de notre projet, nous ne disposons pas d'annotation des régions d'intérêt dans les images (voir section 2.2.2). Ce chapitre propose une solution pour l'apprentissage de régions d'intérêts dans les images, de manière non supervisé et ne nécessitant pas d'annotation des régions au préalable et donc moins coûteux.

5.1 CARTE D'ACTIVATION

Des solutions à base de réseaux à proposition de régions (RPN pour Region Pooling Network) permettent une proposition automatique des régions par le réseaux. Les méthodes de l'état de l'art utilisant ce genre d'approche (section 3.4) obtiennent une segmentation précise de l'image et une amélioration de la précision des plongements [86]. Ces genres

de méthodes nécessite toutefois un ensemble d'images annotées avec leurs régions pour fonctionner. Nous ne disposons pas d'une base de donnée annotées au niveau des régions. Ceci nous amène à développer des solutions ne nécessitant pas la présence de boîtes englobantes dans le corpus d'apprentissage. Nous avons vu dans la section précédente qu'un apprentissage fin est nécessaire (étape 1 et 2 de la figure 4.5). Ceci ne donne pas en soit des résultats suffisants pour l'identification d'instance, mais permet d'apprendre au réseau une certaine représentation des objets.

Le réseau est entraîné à reconnaître certains objets, et la sortie du réseau est dépendante de la présence ou non de ces objets dans les images. Il a été montré [82] que lorsqu'un réseau est appris sur une collection, il est capable d'identifier la présence de certains objets, même s'il ne les classe pas explicitement. Ce qui signifie que si l'on applique notre réseau sur chaque sous-partie de l'image, on obtient des maximums d'activation du réseau, si un des objets qu'il a appris à reconnaître est présent. On prend les zones où le réseau a des activations plus importantes, on peut créer une carte des activations du réseau. Ces cartes d'activations sont présentées sur la figure 5.1. Un réseau de type ResNet, après un apprentissage fin sur la collection, est appliqué sur un certain nombre de sous-parties de l'image. Les zones en rouge sur les cartes d'activations représentent les endroits où le maximum d'activation était le plus important. On remarque que ces zones correspondent aux emplacements des objets, avec quelques erreurs sur le troisième tableau. On voit dans la troisième colonne la classification de chacune de ces sous-parties. Cette classification n'est pas correcte, avec des labels erronés, et très variés. Ces résultats sont en relation avec les résultats relativement faibles obtenus dans le chapitre précédent avec la classification, 79% contre 92% pour l'extraction de caractéristiques. En revanche, cette méthode permet d'identifier les régions d'intérêt de l'image.

Le problème avec l'application d'un réseau de cette manière est le coût de calcul. Par exemple, temps de passage d'une image $224 * 224$ à travers AlexNet est de 1.2 ms. Si on applique ce même réseau sur une image $448 * 448$ avec un pas de 56 pour créer une carte d'activation de $8 * 8$, il faut 76.8 ms.

Cette méthode n'est donc pas envisageable, surtout si l'on souhaite utiliser des images plus grandes pour obtenir une carte d'activation plus précise. La solution est d'utiliser des réseaux entièrement convolutifs. Dans la section 3.4, nous avons vu que les réseaux entièrement convolutifs pouvaient être utilisés pour la segmentation d'image. Dans le cas où nous n'avons pas de base de données d'apprentissage de la segmentation, nous entraînons le réseau comme expliqué précédemment, et nous remplaçons les couches entièrement connectées par une couche de convolution. Il y a une équivalence entre une couche entière

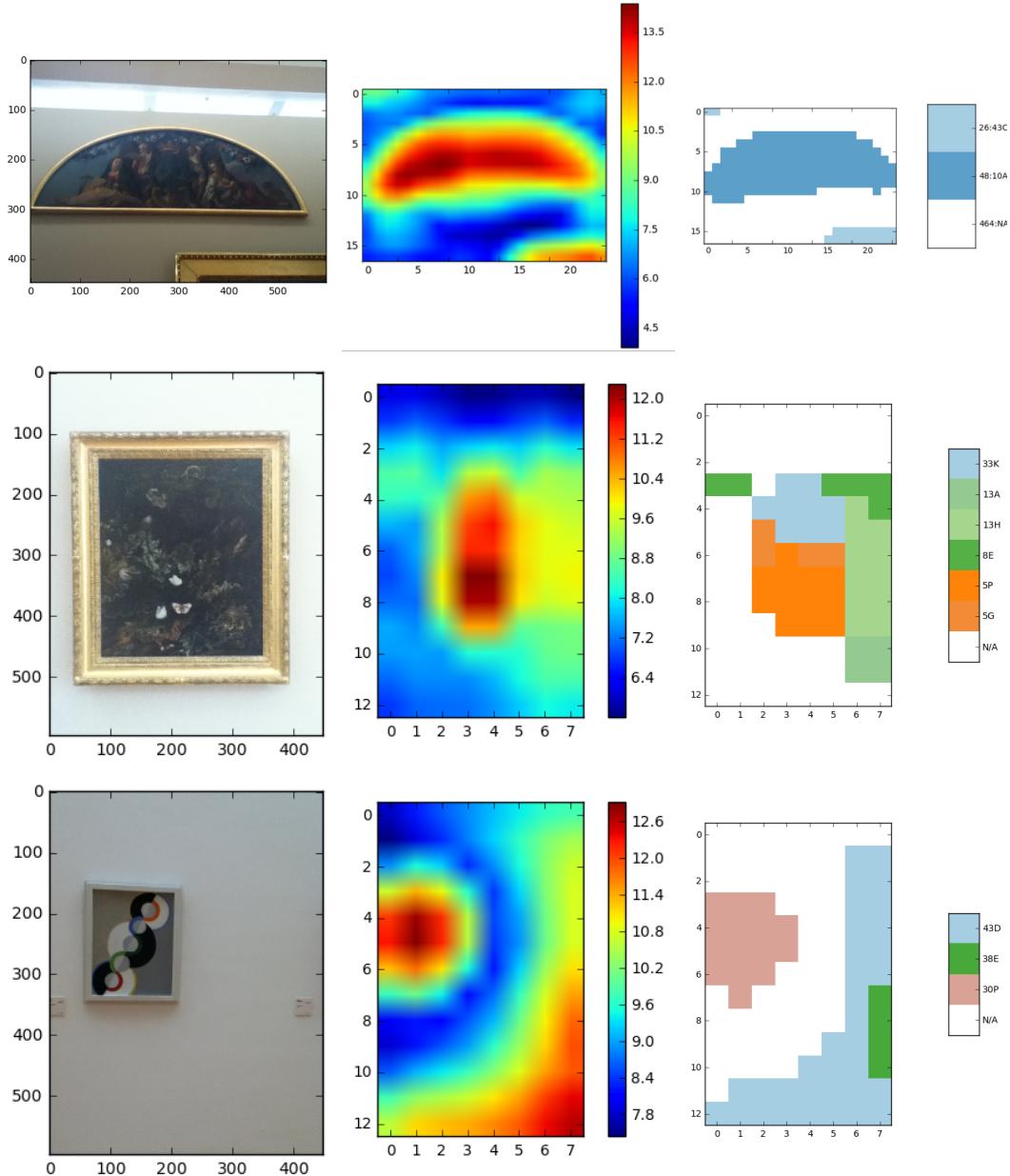


Figure 5.1 : Exemple d'images avec les heat-map des activations maximales, obtenues à partir d'un ResNet-152 après un apprentissage fin. Le réseau est appliqué sur toute l'image de manière strié. Les labels obtenus sur les zones d'activation maximales sont également indiqués.

rement connectée et une couche de convolution 1×1 , comme montré sur le schéma 5.2, avec le même nombre de paramètres. L'avantage de la couche de convolution est que l'on peut l'appliquer sur n'importe quelle taille d'image (figure 5.2c), la taille de sortie dépendant de la taille d'entrée. Le temps d'exécution d'un réseau type AlexNet entièrement convolutionnel sur une image de 448×448 est de 18 ms, soit 5 fois plus rapide que l'approche

naïve.

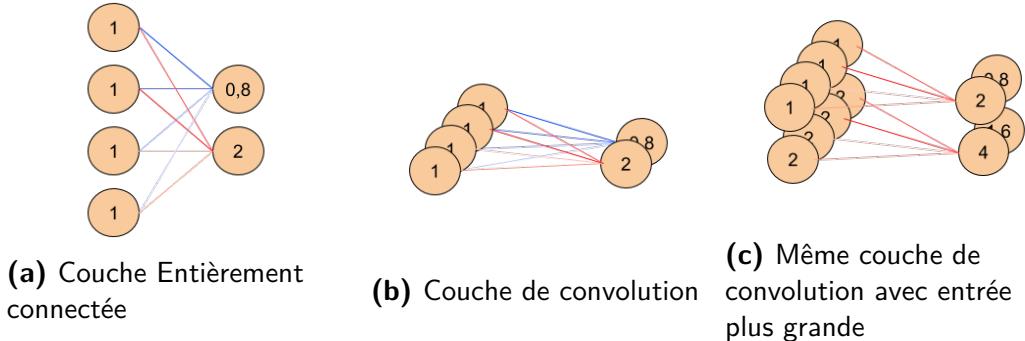


Figure 5.2 : Exemple d'équivalence entre une couche entièrement connectée et une couche de convolution. Le nombre de paramètres est le même, ici $4 * 2$, ils peuvent donc être copié pour obtenir les mêmes résultats. On peut appliquer la convolution avec une entrée plus grande, en une seule passe.

5.2 APPRENTISSAGE SUR DIFFÉRENTES RÉGIONS

L'approche présentée précédemment permet de mettre en avant qu'il est possible de détecter les régions sans apprentissage direct de celles-ci. Nous pouvons toutefois améliorer les résultats de la détection de régions d'intérêt en faisant un apprentissage fin sur des régions spécifiques de l'image, plutôt que sur l'image entière. En utilisant différentes échelles et différentes régions, toujours pour apprendre la même classe, nous pouvons forcer le réseau à détecter les régions dans l'image. La fonction de coût associé à cette apprentissage est la moyenne de l'entropie croisée entre les régions et entre les échelles. Dans le cas de la classification, l'étiquette d'une image, qui représente sa classe, est un vecteur \hat{x} de dimension C , où C est le nombre de classes, où chaque éléments \hat{x}_i est à zéro si la classe i n'est pas présente, 1 sinon. La sortie du réseau de neurones x est une distribution de probabilité pour chaque classe. L'entropie croisée E est définie par :

$$E(x, \hat{x}) = - \sum_{i=1}^C \hat{x}_i \log(x_i) \quad (5.1)$$

Pour une image à une échelle s donnée, définissons H_s et W_s comme respectivement la hauteur et la largeur de la carte de caractéristiques. Chaque pixel de la carte d'activation correspond à un sortie du réseau de neurones pour une région de l'image donnée. Le nombre de région est donc $H_s * W_s$. Nous notons $x_{h,w}$ la distribution de probabilité (la sortie du

réseau) sur la région (h, w) . Nous définissons la fonction objectif suivante :

$$\mathcal{L}(x, \hat{x}) = \frac{1}{S} \sum_{s=1}^S \frac{1}{H_s * W_s} \sum_{h=1}^{H_s} \sum_{w=1}^{W_s} E(x_{h,w}, \hat{x}_{h,w}) \quad (5.2)$$

qui représente la moyenne de l'entropie croisée de l'ensemble des régions, moyennée sur l'ensemble des échelles de l'image. La figure 5.3 représente l'apprentissage avec cette fonction objectif. L'idée étant de forcer le réseau de neurones à se concentrer sur certaines régions, pour classifier l'image.

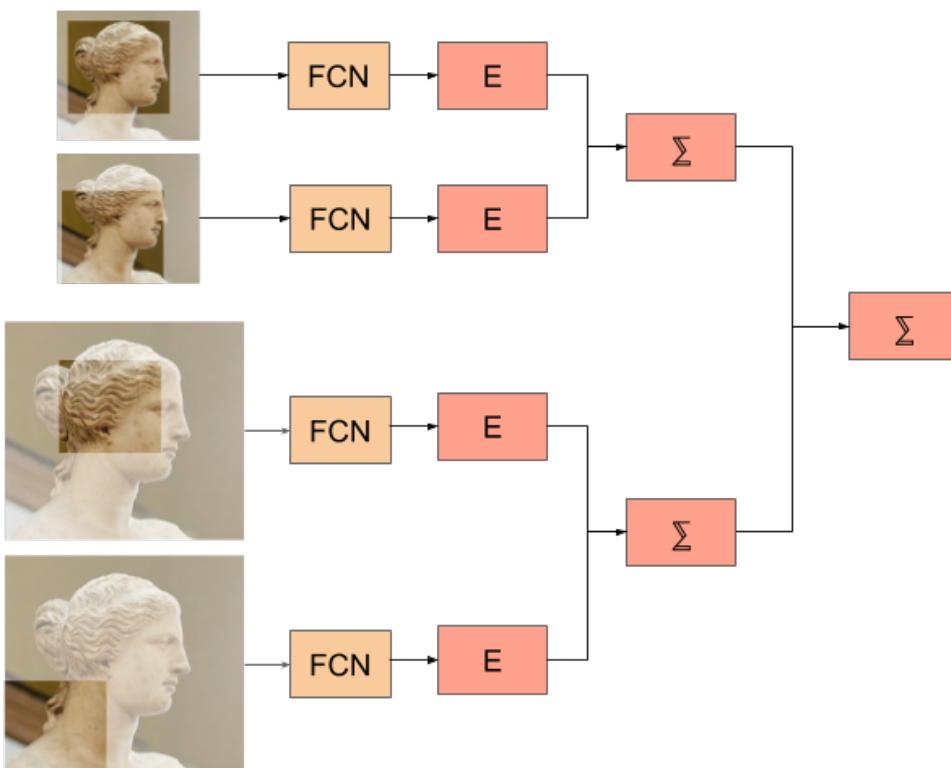


Figure 5.3 : Calcul de la fonction de coût lorsque l'on entraîne le réseau sur différentes régions à différentes échelles. L'entropie croisée E est calculé pour chaque segment d'image avant d'être moyenné (équation. 5.2).

5.3 APPRENTISSAGE DE RÉGIONS ET DE SIMILARITÉ

Une fois l'apprentissage sur les régions réalisé, nous disposons d'un réseau capable de se focaliser sur une certaine région de l'image. Nous utilisons ceci pour changer la fonction

objectif triple. Celle définie précédemment (équation 4.5) ne prend pas en compte les régions. Si l'on prend en compte comme région d'intérêt les k éléments avec la plus forte activation du réseau, on peut extraire les caractéristiques depuis uniquement cette zone, qui représente la région la plus probable de présence de l'objet. Pour la fonction objectif de l'apprentissage, en plus du plongement correct des images positives et négatives, nous ajoutons la classification de cette région avec l'entropie croisée. Ce qui donne l'équation suivante :

$$\mathcal{L}(x, y, z, \hat{x}) = \alpha \max(0, x \cdot z - x \cdot y + m) + (1 - \alpha) \frac{1}{k} \sum_{l=1}^k E(x_{h,w}, \hat{x}_{h,w}) \quad (5.3)$$

On retrouve le plongement des images x , y et z , qui doivent vérifier l'objectif triple (équation 4.5). Ainsi que la classification des k régions de plus forte activation, avec α la régularisation entre les deux objectifs de la fonction de coût.

Dans le but de réaliser cette fonction, nous définissons un réseau de neurones entièrement convolutif, capable de produire une sortie de classification de la région de plus grande activation, ainsi qu'un plongement de cette région. Pour ce faire, une couche de “pooling” des k régions de plus forte activation est ajoutée à la sortie du réseau. En s'inspirant de l'architecture R-MAC [86], une normalization L_2 , un décalage et une couche entièrement connectée sont ajoutés à cette sortie. Ce pipeline permet de réaliser une PCA (Principal Component Analysis) pour réduire la taille de la représentation [137]. Les paramètres du décalage sont appris par rétro-propagation et la couche entièrement connectée permet de réduire la taille du(descripteur à la taille désirée pour l'espace de projection \mathbb{E} . Une normalisation L_2 est de nouveau opérée pour normaliser le plongement des images.

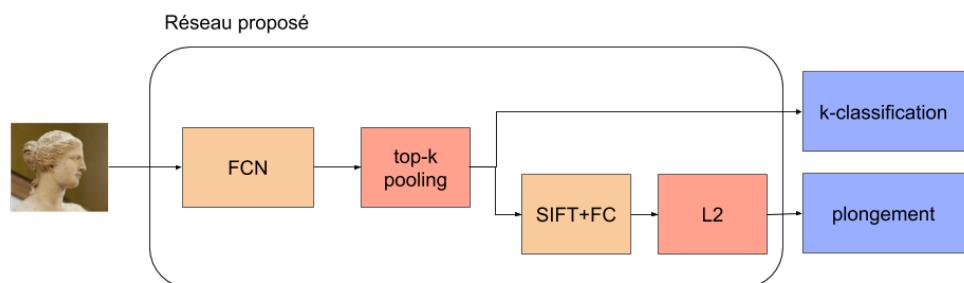


Figure 5.4 : Architecture du réseau proposé basé sur un réseau entièrement connecté pré-entraîné pour détecter les régions d'intérêt.

Pour l'entraînement de ce réseau, nous utilisons une architecture à trois branches comme

précédemment, à laquelle nous ajoutons la classification des k régions de plus fortes activation, comme montré sur le schéma 5.5. La même stratégie de sélection des triplets que celle utilisée précédemment est appliquée.

La figure 5.5 reprend l'ensemble des étapes de l'apprentissage, en incorporant l'apprentissage des régions. Les étapes 1 et 2 sont identiques à celles présenté au chapitre précédent. L'étape 3 correspond à l'apprentissage des régions d'intérêt. Enfin, l'étape 4 est l'apprentissage grâce à la fonction de coût 5.3.

5.4 EXPÉRIMENTATION

Nous testons notre approche sur deux collections du projet GUIMUTEIC, CLICIDE et GaRoFou, et nous évaluons grâce au métriques suivantes :

- La précision à 1 (TOP@1) : l'image la plus proche contient le même objet.
- La précision moyenne (MAP pour Mean Average Precision) : la moyenne des valeurs de précision des images pertinentes (contenant le même objet) dans la liste ordonnée en fonction des distances.

5.4.1 PARAMÈTRES D'APPRENTISSAGE

Pour les premières étapes de l'apprentissage, les mêmes paramètres que dans le chapitre précédent sont utilisés, comme les étapes 1 et 2 sont identiques. La sélection des triplets se fait également de la même manière que précédemment, à savoir : dans un premier temps tous les triplets négatifs semi-difficiles, puis tous les triplets négatifs difficiles.

Comme nous utilisons un réseau entièrement convolutif, la taille des images pour l'apprentissage est libre, contrairement à un réseau contenant des couches entièrement connectées. Pour nos expérimentation, nous utilisons une taille fixe pour la plus petit côté des images en entrée, avec deux échelles différentes. Pour ResNet, nous utilisons deux échelles : 448 et 224. Pour AlexNet, nous avons trouvé que la convergence n'était pas correcte avec les mêmes paramètres, nous utilisons donc les échelles : 384 et 224. Lorsque l'on utilise des images avec des ratio entre hauteur et largeur très grand, l'utilisation de mémoire pendant l'apprentissage peut être très importante. Pour limiter cela, nous limitons le ratio au maximum à 2.0.

Les paramètres de la fonction de coût 5.3 sont $\alpha = 0,5$ et $k = 6$.

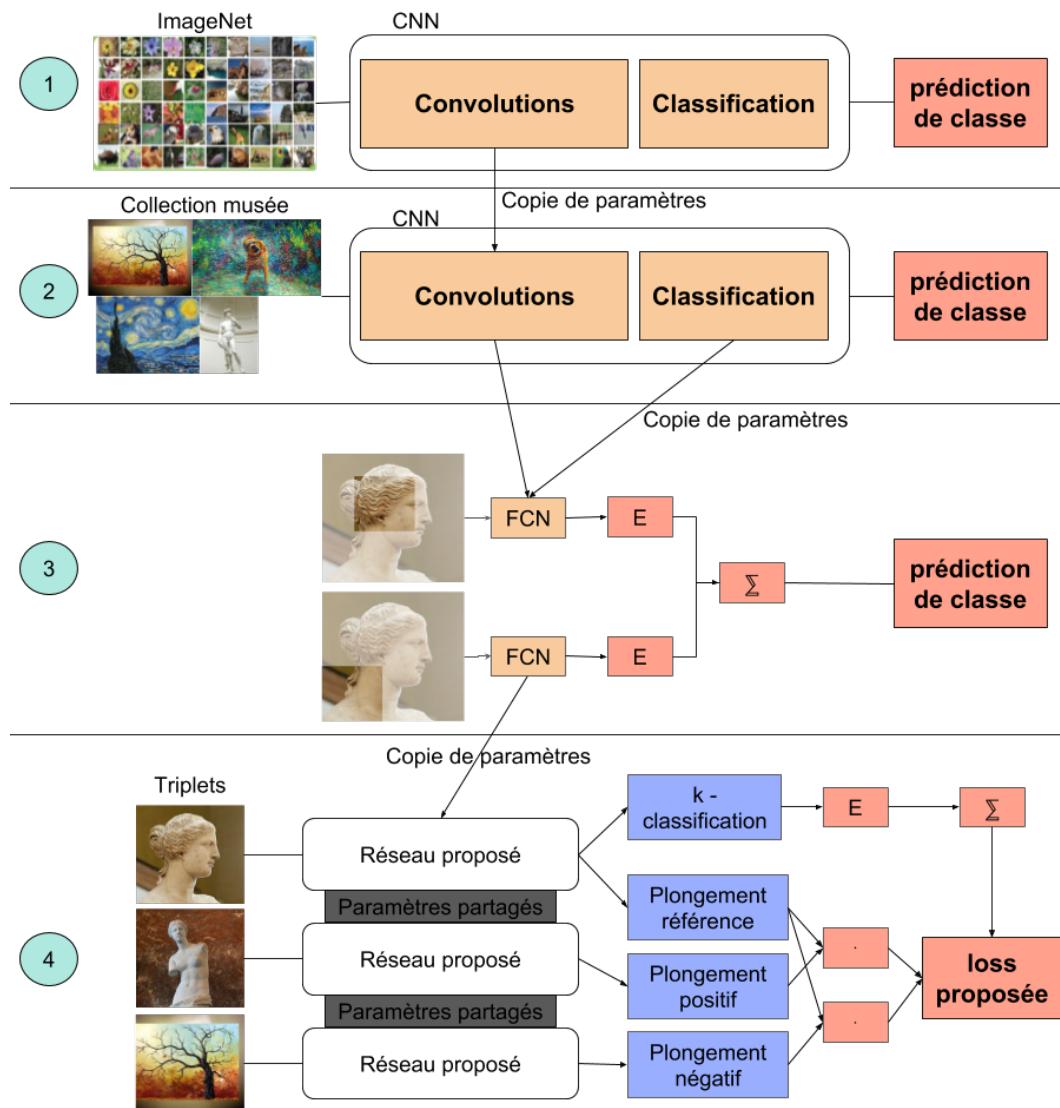


Figure 5.5 : Pipeline final d'apprentissage, avec l'ajout de la détection des régions d'intérêts.

	<i>top@1 (en %)</i>		<i>MAP (en %)</i>	
	<i>CLICIDE</i>	<i>GaRoFou</i>	<i>CLICIDE</i>	<i>GaRoFou</i>
<i>Gordo multi-res [86]</i>	92.73	95.65	65.49	89.32
<i>AlexNet E</i>	72.73	85.87	32.71	66.11
<i>AlexNet Classif</i>	78.18	90.76	38.51	72.92
<i>AlexNet SS</i>	75.76	90.20	36.20	77.73
<i>AlexNet + Régions</i>	81.21	83.15	45.53	71.71
<i>ResNet E</i>	72.12	85.33	40.99	70.15
<i>ResNet Classif</i>	79.39	94.57	75.11	93.44
<i>ResNet SS</i>	85.45	95.11	83.00	91.90
<i>ResNet + Régions</i>	94.55	96.20	82.94	91.83

Table 5.1 : TOP@1 et MAP des différentes approches sur les collections CLICIDE et GaRoFou.

5.4.2 RÉSULTATS

Le tableau 5.1 présente les résultats obtenu sur CLICIDE et GaRoFou par l'approche proposée, comparé aux approches précédentes. Nous remarquons que notre méthode obtient le meilleur TOP@1, avec un score de 94.55% comparé aux 92.73% de l'état de l'art sur CLICIDE. Sur GaRoFou, les méthodes de l'état de l'art obtiennent un score de 95.65%, le gain est moins important, avec 0.55%. Comme montré précédemment, ResNet présente toujours de meilleurs résultats que AlexNet. L'apport de la propositions de régions par le réseau, même sans base d'apprentissage annotées, permet donc d'améliorer les résultats. Toutefois, si l'on s'intéresse à la MAP, on voit qu'aucune des méthodes proposées ne permet d'obtenir de meilleurs résultats qu'un réseau Siamois Simple (SS), où qu'un fine-tuning dans le cas de GaRoFou (Classif). Cela nous amène à penser que même si le TOP@1 est bon, l'espace de projection créé ne capture pas complètement la similarité entre les images.

5.5 INDEXATION ET AUGMENTATION DE DONNÉES CÔTÉ BASE DE DONNÉES

Pour l'identification d'instance, nous utilisons la recherche du plus proche voisin dans la base de données d'image. Un parcours simple de cette base de donnée est suffisant, étant donné la taille de nos collection. On peut même augmenter le nombre d'éléments dans la base de données, ce qui est nommé “Database-side feature augmentation” [108, 138]. Cela permet d'améliorer la recherche, en couvrant plus l'espace de recherche.

Pour modifier les éléments dans l'espace de projection, il y a deux stratégies : soit ajouter des points dans l'espace, soit de changer le plongement de chaque élément de la base de données. On peut par exemple remplacer chaque éléments de l'espace par une combinai-

	TOP@1 (en %)		MAP (en %)	
	CLICIDE	GaRoFou	CLICIDE	GaRoFou
AlexNet proposé	81.21	83.15	45.53	71.71
AlexNet proposé + Augmentation	80.61	82.61	71.02	81.66
ResNet proposé	94.55	96.20	82.94	91.83
ResNet proposé + Augmentation	93.94	95.11	94.23	93.86

Table 5.2 : TOP@1 et MAP sur les corpus CLICIDE et GaRoFou, avec et sans l'augmentation de données coté base de données.

son de ces k plus proches voisins, ce qui permet de lisser les projections, et de corriger le bruit [86]. Dans notre cas, nous disposons généralement de peu d'image références pour chaque objet, nous n'envisageons donc pas de modifier les plongement en fonction de leur voisin, mais plutôt de créer de nouvelles projections, pour rendre l'espace plus dense.

Pour l'ensemble des exemples d'une instance, nous prenons chacun des plongements, et calculons leur moyenne. Ce nouveau vecteur peut être considéré comme un nouvel exemple pour l'instance donnée. Nous pouvons donc l'ajouter à l'espace de projection. Ce nouveau point peut également servir de référence pour l'instance qu'il représente, ce qui permet de supprimer tous les autres éléments de l'espace. Ainsi, le nombre d'éléments à parcourir pour la recherche du plus proche voisin n'est plus l'ensemble des exemple, mais uniquement un pour chaque instance.

Le tableau 5.2 montre les résultats obtenu avec l'augmentation de données de la base de donnée par rapport au résultats précédents. L'ajout de l'augmentation de donnée n'améliore pas le top@1, voir même la détériore, avec en moyenne une perte de 0.5%. Par contre, on remarque une nette amélioration de la MAP, ce qui signifie que les nouveaux exemples créé dans la base de donnée capture bien certaines instances.

La figure 5.6 montre quelques exemples de succès et d'échecs de notre approche. Pour chaque requêtes (colonne de gauche), on affiche les deux images les plus proches. On remarque que dans les échecs, la deuxième image retournée est la bonne (la TOP@2 est de 100%). L'augmentation de données dans l'espace de projection permet donc de créer un espace plus dense, qui capture mieux la similarité entre les images. Même si cela n'améliore pas la TOP@1 dans notre cas, cela permet d'envisager d'autre approches pour la détection d'instance, notamment si l'on s'intéresse aux vidéos, avec par exemple un vote majoritaire ou une moyenne des instances retournées.

Notre approche permet donc d'obtenir de meilleurs résultats que les méthodes de l'état de l'art, dans le cas des corpus de petites tailles. Notre solution d'apprentissage de régions

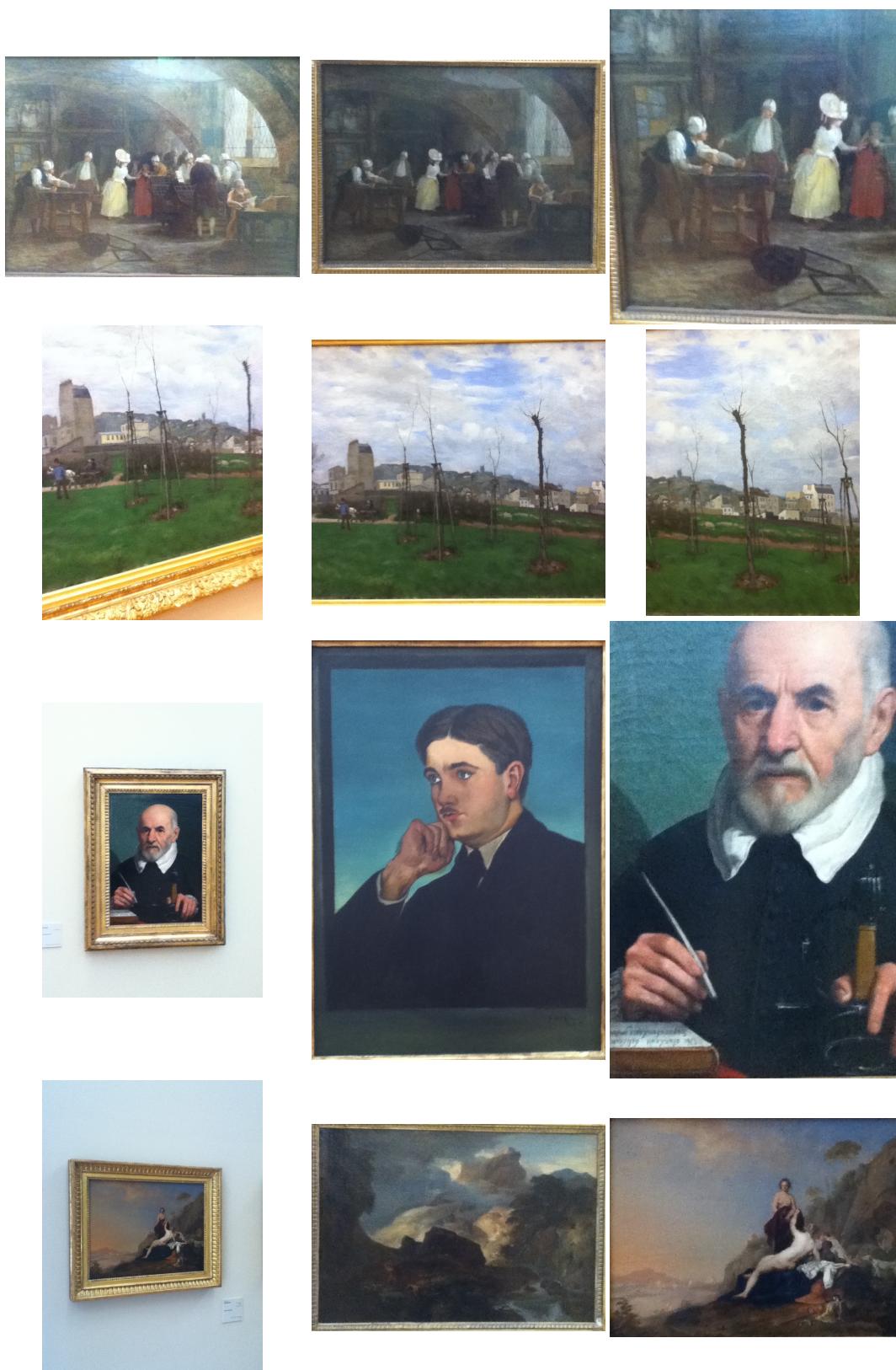


Figure 5.6 : Exemples de succès et d'échecs de notre approche. La première colonne représente les requêtes, les deuxièmes colonnes contiennent les images les plus proches et la troisième colonne la deuxième image la plus proche dans la collection CLICIDE.

non supervisé permet d'avoir une proposition de régions d'intérêts par le réseau. La proposition de régions pour la création du plongement semble être un point clef pour la recherche d'instance dans les images. Enfin, une densification de l'espace de projection par augmentation de données permet de mieux capturer la similarité entre les images dans l'espace de projection, avec cependant une perte en précision.

6

Détection de gestes

Nous avons proposé, dans le chapitre précédent, une nouvelle approche pour l'identification d'instance sur les images. Nous nous intéressons dans ce chapitre à l'interaction avec l'utilisateur, à travers ses gestes. L'étude participatives détaillée dans le chapitre 2 intitulé "Les enjeux de l'interaction dans les visites touristiques", en annexe B, a mis en évidence 5 gestes utiles pour l'interaction entre l'utilisateur et le système GUIMUTEIC. L'objectif de ce chapitre est de présenter une solution pour la détection de gestes dans le cadre du projet, basé sur la caméra embarquée.

Nous avons pour contrainte de faire fonctionner cette détection de geste sur appareil mobile, en continu, pour être capable de réagir aux actions de l'utilisateur. L'objectif est double : avoir une bonne reconnaissance des gestes, avec un minimum de calculs pour être capable de fonctionner sur processeur mobile. Nous nous intéressons dans un premier temps aux réseaux de petites tailles et aux temps de calculs faible, pour proposer un réseau de neurones profond performant et compact (section 6.1). Nous étendons ensuite cette nouvelle architecture pour qu'elle soit capable de regarder plusieurs trames de la vidéos. En ajoutant une fusion de l'information venant de plusieurs trames, nous obtenons une solution efficace, ayant les mêmes performances que les architectures de l'état de l'art, avec considérablement moins de paramètres (section 6.2). Nous évaluons nos propositions sur un corpus de vidéos de gestes conçu dans le cadre du projet GUIMUTEIC, présenté en

détail en annexe A.5.

6.1 UTILISATION DES CONVOLUTIONS GROUPÉES POUR RÉDUIRE LA COMPLEXITÉ

Notre objectif est de proposer une architecture de réseau qui soit à la fois assez compacte et rapide pour être utilisable sur processeur mobile, et suffisamment performante pour être au niveau de l'état de l'art. Nous avons présenté dans la section 3.3, un état de l'art des réseaux de petites tailles, et nous nous basons sur ceux-ci pour créer notre architecture. Les réseaux de petites tailles tels que MobileNet [20], SqueezeNet [8] ou ShuffleNet [10] utilisent les principes présentés par Simonyan et Zisserman [5]. Le premier étant de n'utiliser que des convolutions 3×3 , ou équivalents. Ce type de convolutions étant coûteux en paramètres et en temps de calcul, nous proposons de les remplacer par deux convolutions avec moins de paramètres (section 6.1.1). Le deuxième principe à respecter est d'utiliser deux types de convolutions en alternance : un premier ne modifie pas la taille des entrées et un deuxième qui diminue la taille des entrées, mais augmente le nombre de canaux. En enchaînant ces deux types de blocs, on peut construire des réseaux très profonds, tout en conservant le plus d'informations le long du réseau. Nous proposons deux nouveaux blocs S1 (section 6.1.2) et S2 (section 6.1.3) qui correspondent à ces critères avec un nombre de paramètres réduits.

6.1.1 REMPLACER LES CONVOLUTIONS 3×3

Les convolutions 3×3 sont, depuis VGG [5], quasiment les seules convolutions utilisées dans les réseaux de neurones. En combinant plusieurs, il est possible de réaliser des convolutions de n'importe quelle taille, avec l'avantage de pouvoir mettre entre chaque convolution une opération non linéaire (neurones ReLU) ou des opérations de régularisation (*Batch-Norm*), qui améliorent la convergence de réseau et sa généralisation. Cependant, dans le cas des réseaux compacts, les convolutions représentent jusqu'à 89% du total des calculs fait par le réseau [139].

Pour remplacer les convolutions 3×3 , nous utilisons une combinaison d'une convolution 3×3 *DeepWise*, présentée dans la section 3.3.3, et une convolution 1×1 . La convolution 3×3 *DeepWise* (DW-Conv 3×3) est une convolution où chaque canal de sortie n'est connecté qu'à un seul canal d'entrée. La convolution 1×1 ne s'intéresse qu'à un pixel, sur chacun des canaux. On obtient ainsi une information au niveau du voisinage du pixel avec la première convolution, et une fusion des informations des différents canaux grâce à la deuxième. En

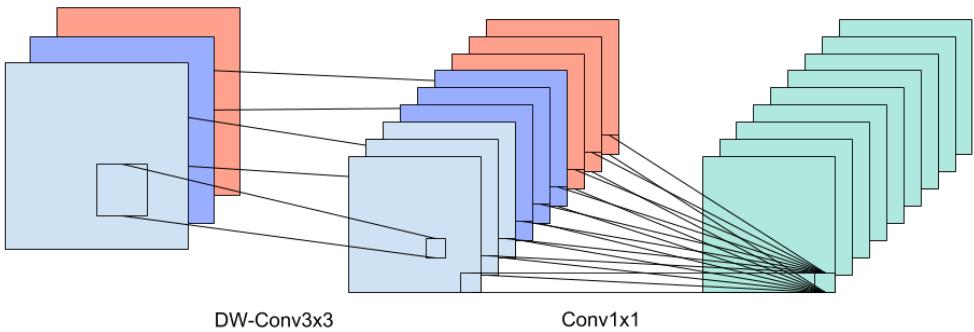


Figure 6.1 : Convolution 3×3 DeepWise suivie d'une convolution 1×1 qui fait office de fusion.

combinant les deux comme sur la figure 6.1, nous obtenons une opération proche d'une convolution 3×3 au niveau des performances [20], avec moins de paramètres.

Au niveau de la complexité, une convolution 3×3 sur 32 canaux d'entrée, avec 32 canaux de sortie, contient $(3 \times 3) * 32 * 32 + 32 = 9248$ paramètres. Une convolution 3×3 DeepWise contient pour sa part uniquement $(3 \times 3) * 32 + 32 = 320$ paramètres, comme on supprime toutes les connexions entre les canaux, soit 32 fois moins de connexion. La convolution 1×1 toujours avec 32 canaux contient $(1 \times 1) * 32 * 32 + 32 = 1056$ paramètres. On obtient donc au final en combinant les deux $320 + 1056 = 1376$ paramètres, ce qui représente une diminution de 85% par rapport au 9248 de la convolution 3×3 .

Nous comparons dans le tableau 6.1 les temps de calcul des différentes couches sur un processeur Quad Core cadencé à 2.60GHz, sous Linux 64 bits¹, en utilisant le framework Pytorch². On observe une diminution passant de 23.9 millisecondes pour la convolution 3×3 à 13.1 millisecondes pour la combinaison de la convolution 3×3 DeepWise et de la convolution 1×1 , soit une diminution de 45%. Nous utilisons les temps sur CPU, et non pas sur GPU, car nous nous intéressons au déploiement de notre réseau, et non pas à son entraînement, qui lui est réalisé sur GPU. Pour obtenir des écarts stables et significatifs entre les différents runs, nous utilisons des batchs de 64 images, chacune d'une taille de 32×32 .

Nous avons donc un moyen de remplacer les convolutions 3×3 pour minimiser le nombre de paramètres, et donc l'occupation mémoire, et les temps de calculs. Cependant, nous souhaitons également tirer avantage des *skip-connection* et de la régularisation. Nous pré-

1. Tous les calculs présentés dans la suite sont réalisés sur la même machine

2. <https://pytorch.org>

couche	# paramètres	temps de calcul (ms)
Conv3x3	9248	23.9
Conv3x3G	320	9.94
Conv1x1	1056	4.06
Conv3x3G + Conv1x1	1376	13.1 ³

Table 6.1 : Tableau de comparaison du nombres de paramètres et du temps de calculs des convolutions groupées et non groupées

sentons dans la suite deux nouveaux types de blocs, remplaçant les convolutions 3×3 par ce que nous avons présenté, et ajoutant des connexions et de la régularisation.

6.1.2 BLOC S1

Nous avons énoncé précédemment (section 3.2.2) l'utilité d'avoir deux types de blocs de convolutions, avec l'un qui conserve la taille des entrées que nous nommons S1, et l'autre qui divise par deux la taille des entrées, en multipliant le nombre de canaux, nommé S2. Le bloc S1 que nous proposons est composé de la combinaison des convolutions présentées précédemment. Tout d'abord une convolution 3×3 DeepWise, avec un pas de 1 et un remplissage (padding) de 1, pour conserver la taille des entrées. Ensuite, une convolution 1×1 avec pas de 1 et pas de remplissage, chargée de faire la fusion des informations des différents canaux. Pour améliorer la généralisation de notre réseau, nous ajoutons après chacune de ces couches, une *batch-normalisation*. Nous utilisons également le fait d'avoir remplacé la convolution 3×3 par deux convolutions pour mettre une couche de neurones ReLU entre les deux, ce qui augmente la non-linéarité du bloc. Nous proposons également une *skip-connection* entre l'entrée et la sortie du bloc, pour permettre une meilleur propagation du gradient [6]. La figure 6.2 montre une représentation graphique du bloc S1, et le code qui réalise cette fonction est indiqué en annexe C.1.1.

#couche	type de couche
1	convolution 3×3 DeepWise, pas de 1, remplissage de 1
2	<i>batch-normalisation</i>
3	neurones ReLU
4	convolution 1×1 , pas de 1, pas de remplissage
5	<i>batch-normalisation</i>
6	neurones ReLU
7	fusion des informations entre l'entrée et la sortie par addition

Table 6.2 : Composition du bloc S1.

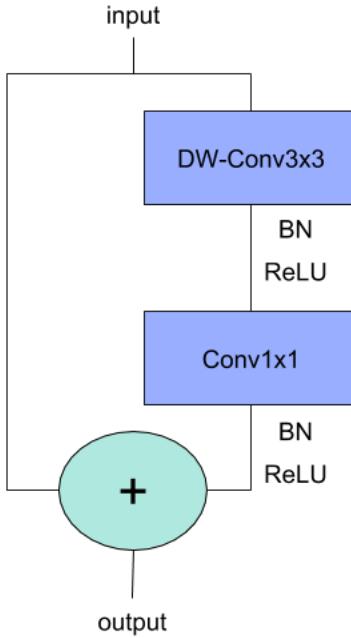


Figure 6.2 : Schéma du module S1.

Les temps de calculs, rapportés dans le tableau 6.3, montrent une large augmentation du temps de calcul, due à l'ajout de la *batch-normalisation* et des couches ReLU. Si on compare au tableau précédent, on passe de 13.1ms à 23.9ms. Cependant, au niveau des paramètres, on ajoute ici uniquement ceux de la *batch-normalisation*, ce qui donne un total 1504. Nous comparons notre bloc aux blocs “Fires” de SqueezeNet [8] et “Shuffle” proposé par ShuffleNet [10] dans le tableau 6.3. Ces deux blocs se basent sur la diminution du nombre de canaux entre les deux couches de convolutions, pour réduire le nombre de canaux d’entrée pour la convolution 3×3 , ce qui permet de réduire le nombre de paramètres. Les deux réseaux doivent utiliser une coûteuse convolution 3×3 en première couche pour extraire les premières informations de l’image. Comme le bloc S1 a pour but de pouvoir remplacer toutes les convolutions, nous pouvons nous passer de cette couche, ce qui réduit le nombre de paramètres du réseau final. On voit dans la tableau que le bloc S1 est plus lent, 23.9ms, que les bloc Fire (22.3ms) et Shuffle (21.8ms).

Notre bloc S1 permet donc de remplacer une convolution 3×3 , avec un nombre de paramètres inférieur et en ajoutant une *skip-connection*. Il est fait pour conserver la taille des tailles des entrées, en ne modifiant ni les tailles des filtres, ni le nombre de canaux. Nous définissons dans la partie suivante le bloc S2, qui peut agir sur le nombre de canaux, et qui va diviser par deux la tailles des filtres d’entrée.

Bloc	# paramètres	temps de calcul (ms)
Conv3x3DW + Conv1x1	1376	13.1
Fire Bloc	2888	22.3
Shuffle Bloc	472	21.8
Bloc S1	1504	23.9

Table 6.3 : Tableau de comparaison des blocs des réseaux de tailles réduites avec le bloc S1

6.1.3 BLOC S2

Pour remplacer les convolutions qui réduisent la taille des entrées, et notamment la première couche 3×3 utilisée par la plupart des réseaux, nous définissons le bloc S2. Nous voulons toujours prendre avantage des *skip-connection*, mais cette fois-ci nous avons une diminution de la taille de l'entrée et un changement du nombre de canaux, il n'est donc pas possible de directement additionner l'entrée et la sortie. Tout d'abord le nombre de canaux étant différent, nous proposons de remplacer l'addition par une concaténation. Ceci permet de garder les apports des *skip-connection*, avec des résultats proches de ceux de l'addition [6, 10, 139]. Cependant, pour pouvoir réaliser la concaténation, chaque canal doit avoir la même taille. Nous proposons donc de fixer le changement de taille à une division par deux de la taille. Ainsi, inspiré par [139], en appliquant un *Average-Pooling* de 3×3 , avec un pas de 2 et un remplissage de 1, on obtient une sortie deux fois plus petite, qu'il est possible de concaténer à la sortie des couches de convolutions.

La partie convolution de ce bloc est très proche de celle du bloc S1, avec uniquement un changement de la première couche. La couche de convolution 3×3 DeepWise est appliquée avec un décalage de 2 et un remplissage de 2, ce qui a pour effet de diminuer la taille des entrées par deux. Le nombre de canal de sortie du bloc est défini comme le double du nombre de canal d'entrée. La bloc S2 (figure 6.3 représente la bloc S2, avec sur la partie gauche l'opération de Pooling pour diminuer la taille des entrées, et sur la droite, les couches de convolutions.

Le nombre de paramètres, observé dans la deuxième colonne du tableau 6.4, ne change pas par rapport au bloc S1. Nous pouvons le comparer au bloc Shuffle avec décalage de 2, qui utilise la concaténation aussi pour faire la fusion entre entrées et sorties. Ce bloc à 848 paramètres, mais un temps d'exécution de 14.9ms, comparé à 13.2ms pour le bloc S2. Le bloc S2 est beaucoup plus rapide que le bloc S1, venant du fait qu'il applique deux fois moins les convolutions 3×3 , étant donné qu'il utilise un décalage de 1.

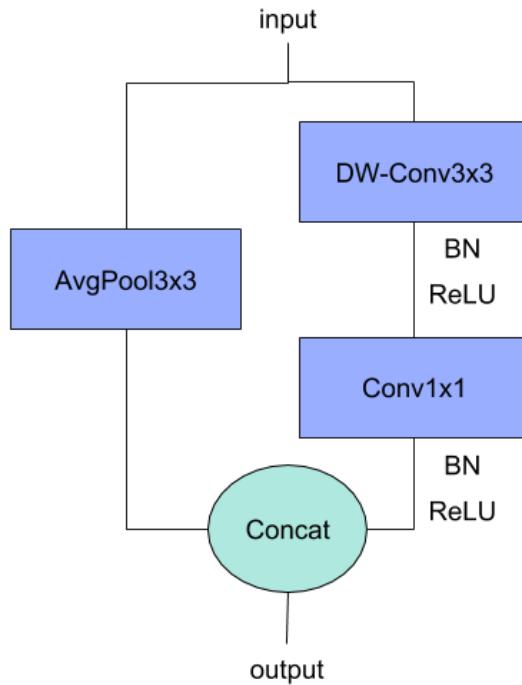


Figure 6.3 : Schéma du module S2.

Bloc	# paramètres	temps de calcul (ms)
Shuffle Bloc Stride 2	848	14.9
Bloc S2	1504	13.2

Table 6.4 : Tableau de comparaison des blocs “Shuffle Stide 2” et S2

6.1.4 RÉSEAU DE CLASSIFICATION DE GESTES À BASE DE S1 ET S2

Dans cette partie, nous construisons un réseau profond en utilisant les blocs S1 et S2. Avec pour objectif d'obtenir un réseau avec un nombre de paramètres faible, et une vitesse d'exécution réduite par rapport aux réseaux profonds de grande taille, comme AlexNet ou ResNet, mais également plus petit que SqueezeNet ou ShuffleNet. Inspiré par les micro-architectures récentes [8, 10, 20, 68] et également par VGG [5], nous proposons de remplacer toutes les convolutions 3×3 par des blocs S1, et toutes les convolutions 3×3 avec décalage de 2, qui font donc du sous échantillonnage, par le bloc S2.

Comme montré sur la figure 6.4, en partant d'une image $224 \times 224 \times 3$, on applique successivement les blocs S1 (sans réduire la taille) et S2 (en réduisant et en augmentant le nombre de canaux, jusqu'à obtenir une dimension suffisante pour la classification, ici $7 \times 7 \times 1024$). Un *Average-Pooling* de 7×7 est alors appliqué, pour obtenir une matrice de $1 \times 1 \times 1024$. Une couche entièrement connectée est alors utilisée pour réduire la dimension

Modèle	# paramètres (en millions)	Temps de calcul (secondes)
SqueezeNet	1.235	0.268
MobileNet	4.231	0.631 ⁴
ShuffleNet	0.922	0.226
Notre modèle	0.772	0.284
AlexNet	61.100	0.380
ResNet	60.192	4.69

Table 6.5 : Comparaison des réseaux de neurones, de petite et de grande taille.

de 1024 à 6, notre nombre de gestes à reconnaître.

Nous comparons notre réseau en terme de taille à des réseaux de petites tailles, tels que SqueezeNet, MobileNet et ShuffleNet, ainsi qu'à des réseaux comme AlexNet et ResNet pour donner une idée des ordres de grandeur. Les résultats de cette comparaison sont présentés dans le tableau 6.5. Pour comparer ces réseaux entre eux, nous nous basons sur le même framework, c'est pourquoi les résultats peuvent différer de ceux présentés dans les publications originales respectives. Les expérimentations sont réalisées avec des batch de 8 images de taille 224x224 à trois canaux.

Nous voyons que notre modèle est celui qui propose le moins de paramètres avec 772 millions de paramètres contre 922 millions, sans être pour autant le plus rapide 0.284 secondes contre 0.226 pour ShuffleNet. Cela venant du fait que nous utilisons plus d'*average-pooling* et de *batch-normalisation* que les autres modèles. L'objectif de notre réseau est de fournir une occupation mémoire la plus faible possible, pour pouvoir être utilisé sur mobile, à côté de notre réseau de reconnaissance d'instance (section 5). En terme de temps de calcul, il est de l'ordre de grandeur des réseaux spécialisés pour les applications mobiles que sont MobileNet, SqueezeNet et ShuffleNet.

Nous n'avons donc pas de problème niveau temps d'exécution, et préférons optimiser l'utilisation de la mémoire, qui est dans notre cas plus problématique, dans le sens où elle doit être partagée avec le réseau de reconnaissance d'œuvres.

Notre réseau, qui nous nommons par la suite le modèle SimpleTrame, est un réseau de petite taille, adapté au processeur mobile, avec un temps d'exécution proche des micro-architectures de l'état de l'art. Dans la partie suivante, nous nous intéressons aux performances de notre architecture dans la tâche de reconnaissance de geste dans les vidéos à la première personne.

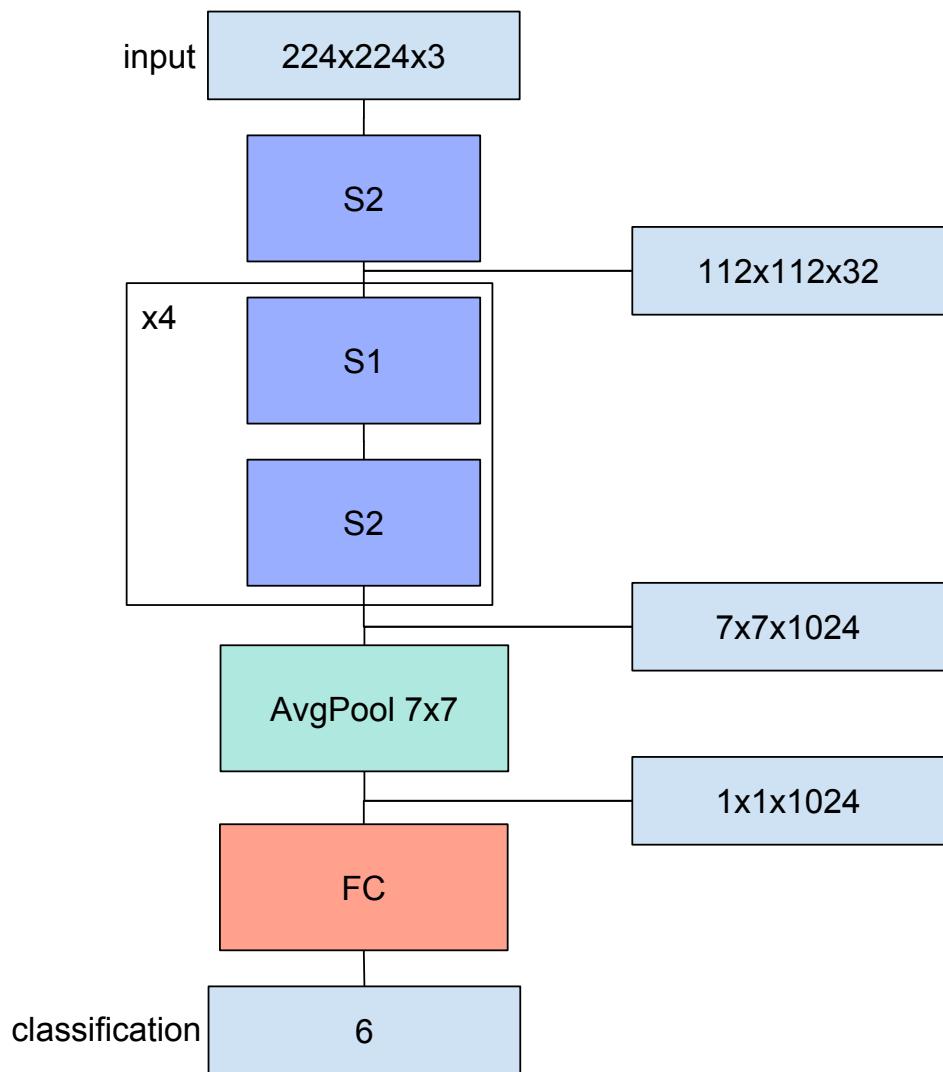


Figure 6.4 : Schéma du réseau proposé pour la reconnaissance de geste, avec entre chaque connection la taille de la matrice de représentation des données.

Modèle	#paramètres (en million)	Précision (en pourcent)
SqueezeNet	1.235	62.50
MobileNet	4.231	56.25
ShuffleNet	0.922	68.75
SimpleTrame	0.772	56.25
AlexNet	61.100	65.62
ResNet	60.192	71.87

Table 6.6 : Comparaison de performance sur trame unique des différents réseaux.

6.1.5 EVALUATION

Nous comparons notre réseau SimpleTrame avec les différentes réseaux de l'état de l'art présenté précédemment, sur la tâche de reconnaissance de geste. Le corpus utilisé est présenté en annexe A.5. Il s'agit d'une collection de vidéo à la première personne, contenant les 6 gestes du projet GUIMUTEIC (5 geste + absence de geste). Il est composé de 29 heures de vidéos, représentant 1669 séquences avec un geste et 1830 séquences sans gestes, ce qui fait un total de 120 000 images. Ce corpus est séparé en trois groupes, 80% de séquences d'entraînement, 10% de séquence de validation et 10% pour le test. Les résultats rapportés dans le tableau 6.6 représentent la précision sur la partie test de cette collection. Pour que la classification d'une séquence soit correcte, il faut que la majorité des images de la séquence soient classifiées correctement par le réseau.

Notre réseau a une précision de 56.25% sur le corpus geste GUIMUTEIC. Ce qui est similaire au résultats de MobileNet, avec moins de paramètres, 0.772 millions contre 4.231. Les réseaux AlexNet, ShuffleNet et ResNet obtiennent de meilleurs résultats, avec un nombre de paramètres également supérieur. Nos résultats sont ici inférieur à l'état de l'art, et ne permettent pas en l'état l'utilisation du réseau dans le cadre du projet GUIMUTEIC. Cependant, ces résultats ne s'intéressent qu'à une seule trame de la vidéo pour la reconnaissance. Dans la section suivante, nous proposons une version sur plusieurs trames de notre réseau, avec une fusion des informations venant de plusieurs instants de la video.

6.2 FUSION D'INFORMATION DE PLUSIEURS TRAMES DE LA VIDÉO

Les gestes à reconnaître peuvent être dynamiques dans la vidéo, et pour identifier correctement la séquence, nous souhaitons tirer profit de l'information provenant de plusieurs trames de la vidéo. Dans la section précédente, nous avons appliqué les réseaux de neurones sur chaque trame des vidéo, en classifiant chacune de ces images. Il est cependant possible qu'un réseau considère plusieurs images. Nous avons présenté dans la section 3.6 les réseau

récursifs. Dans le cas des vidéos, les réseaux récursifs, utilisant les LSTM ou les GRU, sont utilisés sur la sortie d'un réseau convolutif pour réaliser une fusion des informations temporelle. Ils ont l'avantage de ne pas avoir de limite sur le nombre d'images qu'ils peuvent regarder, mais ils ne sont pas adaptés à l'utilisation sur mobile, comme ils requièrent que chaque image soit passée au travers d'un CNN, ce qui peut être coûteux. En utilisant une fenêtre d'observation fixe, il est possible de définir un réseau convolutifs qui s'intéresse à plusieurs images de la vidéo, en réalisant une fusion des informations [16]. Dans cette section, nous nous intéressons à étendre notre réseau pour qu'il capte les informations de plusieurs trames, toujours en limitant le nombre de paramètres.

Nous proposons des nouveaux blocs qui respectent les informations venant de différentes trames (section 6.2.1). Pour réaliser la fusion des informations de plusieurs trames, nous utilisons à une convolution 1×1 , et nous évaluons l'impact de l'emplacement de cette fusion, au niveau performance, mais également au niveau du nombre de paramètres (section 6.2.2).

6.2.1 FRAMEWISE CONVOLUTION

Dans les sections précédentes, nous nous sommes intéressé uniquement à une image, l'entrée d'un réseau était alors une matrice de taille $H * L * C$, où H et L (224 et 224 dans nos expérimentation) sont la hauteur et la largeur des trames, C le nombre de canaux (3 pour RGB). Nous nous intéressons maintenant à un ensemble de trames, donc un certain nombre d'images à la suite les unes des autres. Nous pouvons voir une suite d'image comme une matrice de dimension supérieure $H * L * C * T$, où T est nombre de trame, que l'on peut aussi voir comme une matrice de dimension $3 : H * L * (C * T)$, en concaténant les canaux de chacune des images.

Nos blocs S_1 et S_2 présentés précédemment utilisent les convolutions DeepWise, où chaque canal de sortie n'est connecté qu'à un seul canal d'entrée. Comme chaque convolution ne s'intéresse qu'à un seul canal, en augmentant le nombre de canaux d'entrée d'un facteur T et le nombre de canaux de sortie du même facteur T , la convolution DeepWise est toujours applicable, sans mélanger l'information venant de plusieurs trames. En effet, les C premiers canaux correspondent à l'image 1, les canaux $[C; 2 * C]$ contiennent les informations venant de la deuxième trame, et ainsi de suite.

La fusion des informations entre les canaux est faite dans les blocs S_1 et S_2 par la convolution 1×1 . Pour que celle-ci ne mélange pas les informations venant de plusieurs trames, nous proposons de la remplacer par une convolution 1×1 que nous nommons **FrameWise**. Cette couche doit réaliser la fusion des informations des canaux pour trames, mais pas les informations venant de différentes trames. Pour cela, nous appliquons la convolution par

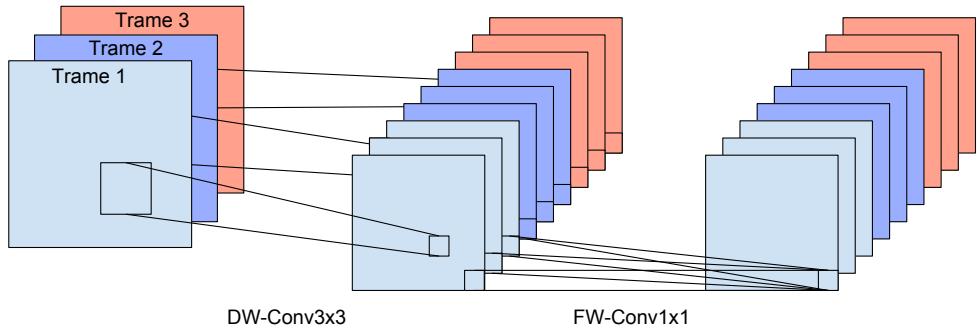


Figure 6.5 : Bloc de convolution utilisant DeepWise Convolution et FrameWise Convolution.

groupe, avec un nombre de groupe égal à T le nombre de trames.

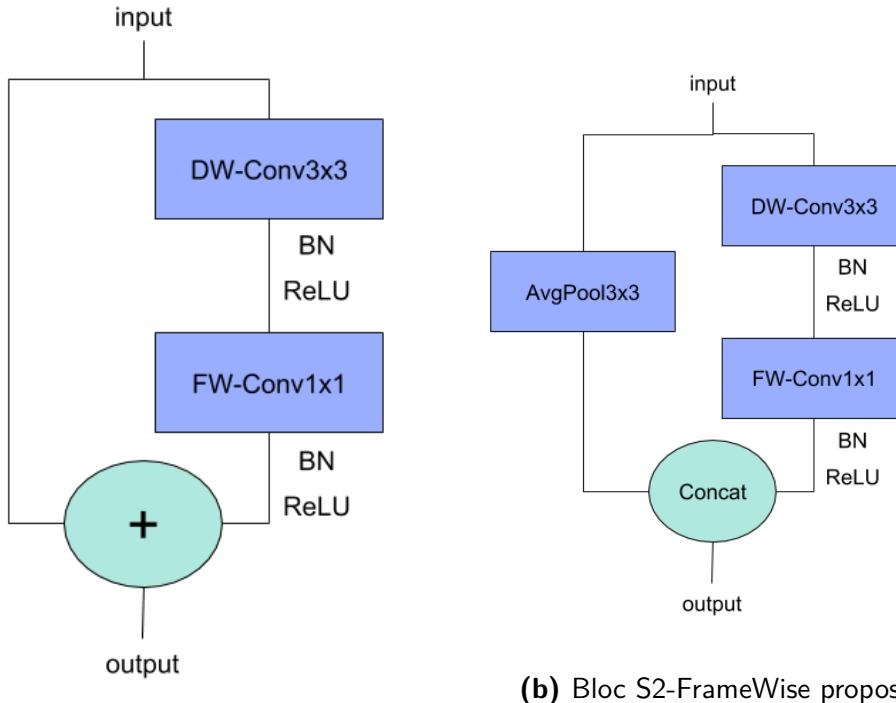
La figure 6.5, montre une version modifiée de la figure 6.1, où les informations venant de chaque trames sont séparées. En entrée, nous avons trois trames, chacune composé de 5 canaux. La convolution 3×3 DeepWise opère canal par canal, et le convolution FrameWise 1×1 réalise une fusion des informations au niveau de chaque trame.

Cela nous permet de définir les nouveaux blocs nommés S₁-FrameWise et S₂-FrameWise, illustrés par le schéma 6.6, en remplaçant uniquement la convolution 1×1 par une convolution FrameWise. Comme on augmente le nombre de convolutions réalisée, le nombre de paramètres va être multiplié par le nombre trames utilisés T .

6.2.2 FUSION D'INFORMATION DE PLUSIEURS TRAMES

Enfin, à un moment dans le réseau, il faut fusionner l'information venant de l'ensemble des trames. Nous définissons donc une couche de fusion, qui sera réalisée par une convolution 1×1 , comme nous avons montré précédemment que ce type de convolution était particulièrement adapté pour fusionner l'information entre les canaux. De plus, à travers plusieurs expérimentation, nous n'avons pas remarqué de différences de précision entre l'utilisation d'un convolution 1×1 et d'une convolution 3×3 . Etant donné que la convolution 1×1 utilise 9 fois moins de paramètres, elle est donc à privilégier.

Nous définissons notre réseau, MultiTrame, qui est très similaire au réseau SimpleTrame 6.4, la seule différence étant l'utilisation de bloc FrameWise, et de la couche de fusion. Sur la figure 6.7, nous montrons une des représentations possible du réseau MultiTrame. Dans cet exemple, la fusion est réalisée au milieu du réseau, couche nommée conv1x1. Avant la fusion, les blocs utilisés sont des blocs FrameWise, on remarque sur la droite du schéma



(a) Bloc S1-FrameWise proposé.

(b) Bloc S2-FrameWise proposé.

Figure 6.6 : Bloc S1 et S2 FrameWise.

que la taille des matrices est plus grande, avec l'ajout du facteur T , le nombre de trames utilisées. Après la fusion, le réseau est similaire au réseau SimpleTrame, avec l'utilisation de bloc S1 et S2.

La fusion peut être réalisée à plusieurs positions dans le réseau. Dans l'exemple sur la figure 6.7, elle est faite après le troisième bloc S2-FrameWise. Cependant, il est possible de modifier le réseau pour que celle-ci soit réalisée plus profondément ou non dans le réseau.

Pour déterminer la position idéale de la fusion, en terme de performance et de nombre de paramètres, nous proposons de faire varier son emplacement dans le réseau. La modification de cette position entraîne un changement du nombre de bloc FrameWise utilisé, et donc du nombre de paramètres. Plus la fusion est réalisée tôt, plus on perd rapidement l'information temporelle, et plus le nombre de paramètre sera réduit. En annexe C.2.1, nous présentons toutes les variantes utilisées pour le test.

Sur la courbe 6.8, nous montrons l'impact de l'emplacement de la fusion sur la précision et sur la taille du réseau. Les numéros 1 à 5 représentent l'emplacement de la fusion, avec 1 pour la première couche et 5 pour la dernière couche. En trait plein, avec la légende à gauche, nous notons la précision de la prédiction du réseau. En trait pointillé, avec la légende à droite,

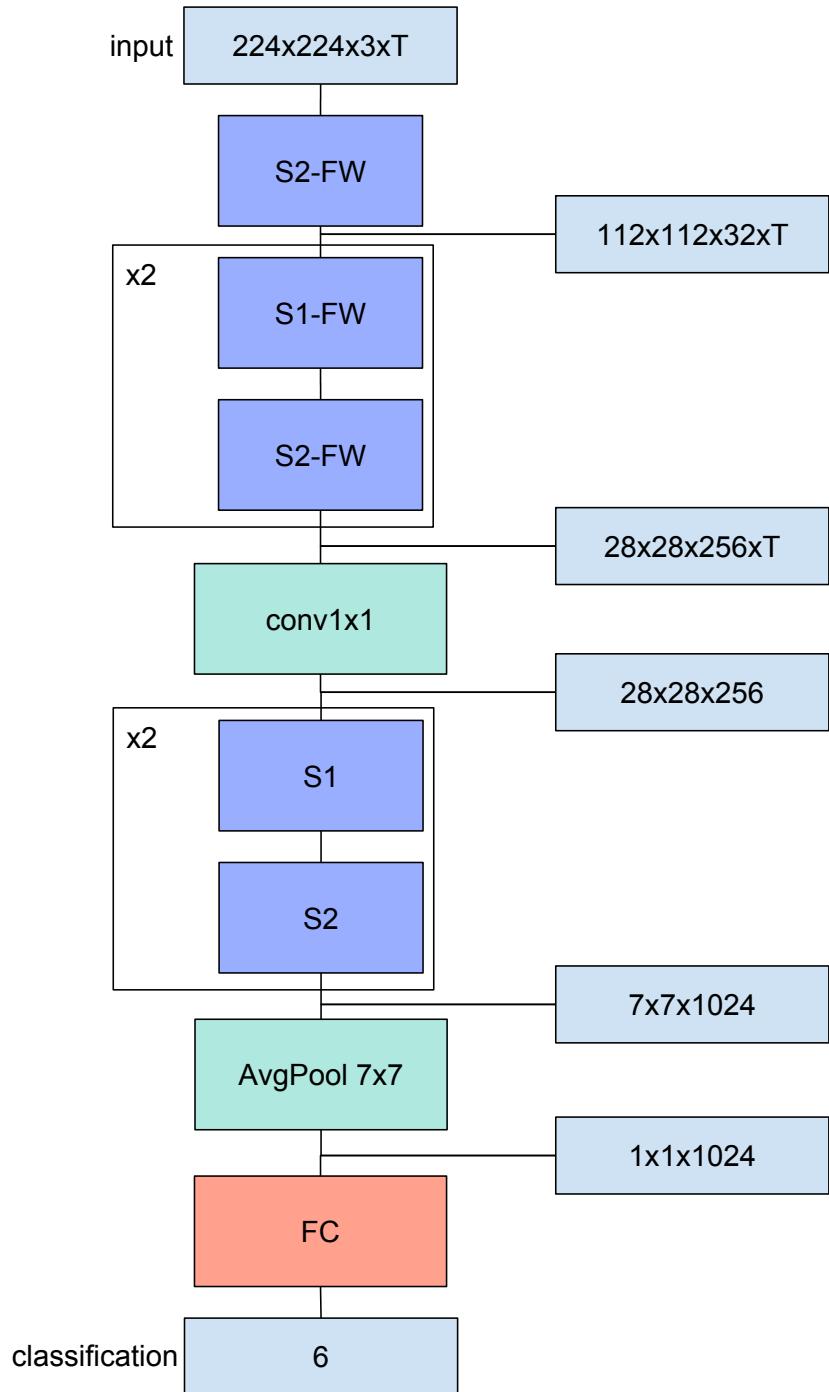


Figure 6.7 : Réseau proposé pour la détection de geste. Les premières couches sont sensibles à la trame (FrameWise), jusqu'à la fusion des N trames par une convolution 1×1 .

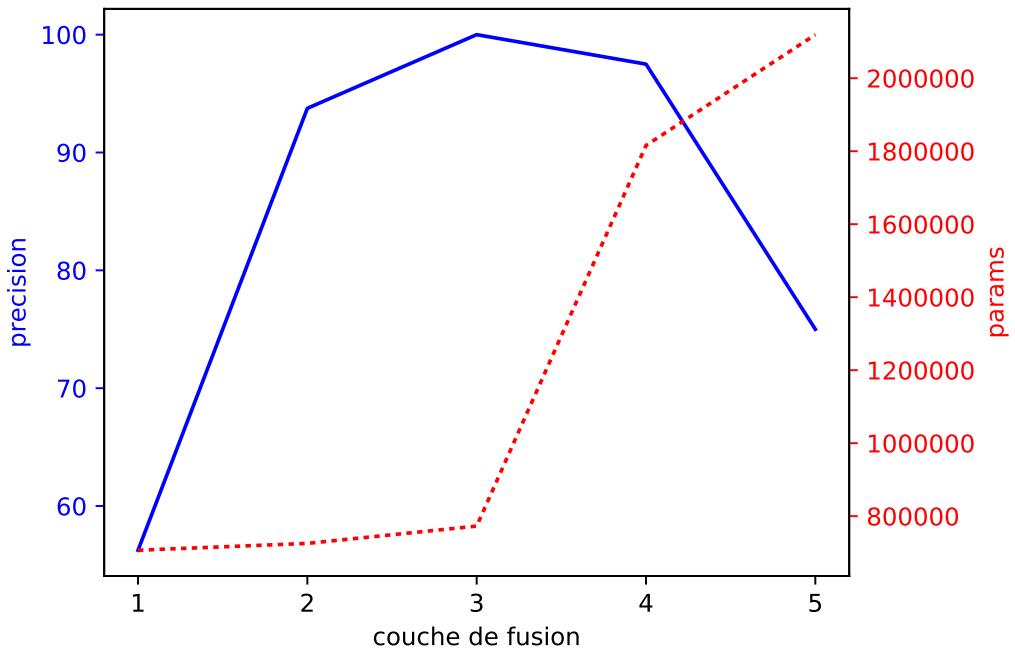


Figure 6.8 : Courbe d'évaluation de la position de la fusion.

nous reportons le nombre de paramètres.

Lorsque la fusion est réalisé sur la première couche, on retrouve la même précision qu'avec le réseau SimpleTrame. Dans ce cas, le réseau est en fait très similaire au réseau SimpleTrame, avec une fusion des informations temporelles réalisée dès le départ, tout le reste du réseau reste inchangé. On remarque un pic de précision lorsque la fusion est réalisée au milieu de réseau, exemple montré dans la figure 6.7. En réalisant les fusion plus tardivement, on remarque une baisse des performances. Cela est dû à un manque de généralisation du réseau, qui est dans un cas de sur-apprentissage. Ceci pourrait être réglé par l'ajout de plus de régularisation ou en modifiant les hyper-paramètres de l'apprentissage, mais on remarque sur la courbe en pointillée la grande augmentation du nombre de paramètres. Nous n'investiguons donc pas ce problème, car nous obtenons une performance optimale avec la fusion au milieu de réseau, et le nombre de paramètres restant proche du réseau simple, cette solution semble optimale.

Dans le tableau 6.7, nous comparons notre réseau au réseau AlexNet et ShuffleNet, avec différentes méthodes de fusion. La fusion 1 se réfère à la concaténation des canaux des différentes images sur la première couche du réseau, sans modifications du reste du réseau. À l'exception d'une légère amélioration dans le cas d'AlexNet, ces résultats sont similaires à ceux obtenu sur trame unique. La fusion EC, correspond à une fusion au niveau de la

Modèle	Fusion	Score (en %)
AlexNet	1	68.75
AlexNet	EC	100.00
AlexNet	LSTM	93.75
ShuffleNet	1	68.75
SqueezeNet	EC	87.50
Notre modèle	1	56.25
Notre modèle	Milieu	100.00
Notre modèle	EC	75.00

Table 6.7 : Tableau de comparaison des différents modèles avec différentes fusion, plus ou moins hautes dans le réseau.

couche entièrement connectée du réseau, c'est-à-dire à la toute fin. La réseau est dans ce cas plus grand, avec un nombre de paramètres multipliés par le nombre de trames, et n'obtient pas toujours les meilleurs résultats, notamment à cause de sur-apprentissage. Pour AlexNet les résultats sont un peu différents, car c'est un réseau plus grand avec des hyper-paramètres bien connus pour éviter le sur-apprentissage. Nous le mettons ici dans le tableau pour montrer que les réseaux profonds classique obtiennent également 100% de reconnaissance sur cette tâche. La fusion au milieu du réseau comme présentée précédemment est la solution qui obtient les meilleurs résultats, avec un nombre de paramètres faible, ce qui correspond à notre objectif.

6.3 CONCLUSION ET LIMITATIONS

Nous avons présenté dans ce chapitre, un nouveau type de bloc pour la constructions de réseau de neurones profonds qui limitent grandement le nombre de paramètres, tout en permettant d'obtenir des résultats proches de ceux de l'état de l'art. Les blocs S₁ et S₂ présentés utilise les convolution DeepWise, connus pour utiliser très peu de paramètres, et les *skip-connection*, qui permettent une meilleur propagation du gradient, et un meilleur apprentissage.

Nous avons également proposé un nouveau type de blocs utilisant des convolutions FrameWise, qui respectent les informations venant de différentes trames de la vidéos. Cela nous permet de définir un réseau qui réalise la fusion des informations des trames en son milieu, ce qui minimise l'augmentation du nombre de paramètres, en permettant d'obtenir les meilleurs résultats de reconnaissance de gestes.

Ces deux blocs, S₁-FW et S₂-FW, nous ont permis de définir un réseau de neurones

utilisant très peu de paramètres, 20% de moins que ShuffleNet. Bien que n'obtenant pas d'aussi bon résultats que ce dernier, notre réseau à l'avantage d'être utilisable en mobilité plus facilement, grâce à une plus petite occupation mémoire. Enfin, utilisé sur plusieurs trames, notre réseau obtient des performances à 100% sur notre corpus de test, au même niveau que les réseaux de l'état de l'art, toujours avec un nombre de paramètres largement inférieur. On en conclut que notre proposition permet bien d'être à l'état de l'art, tout en utilisant moins de paramètres, ce qui facilite son utilisation sur des outils mobiles.

7

Conclusion

Cette thèse s'est intéressée au problème de l'accès à l'information en mobilité, dans le cadre du projet GUIMUTEIC. Ce projet vise à équiper les visiteurs de lieux touristiques avec un audio-guide, équipé d'une caméra pour l'aide à la visite. Les problématiques soulevées par ce projet sont les suivantes :

- Donner accès à de l'information pertinente pendant la visite de manière automatique
- Identifier à quel moment l'utilisateur désire avoir accès à cette information

Des problématiques de déploiement du système viennent s'ajouter à celles-ci, comme le fait d'être sur un appareil mobile, sur lequel doivent fonctionner tous les outils développés dans cette thèse. Il a fallu également déterminer quels sont les gestes utiles pour l'interaction. Pour cela, des séances de conceptions participatives avec des utilisateurs ont été organisées.

Ce travail a donné lieu aux contributions suivantes. Dans le chapitre 4.1, nous avons présenté une méthode pour la reconnaissance d'instances sur des corpus de petite taille. Nous avons pour cela adapté à nos contraintes les systèmes de l'état de l'art utilisant des réseaux siamois à trois branches pour l'apprentissage de similarité entre les images. Nous avons définie une nouvelle fonction objectif, utilisant le produit scalaire pour un calcul rapide de similarité entre les images. Nous avons également proposé une nouvelle méthode de sélec-

tion de triplets pour l'apprentissage, permettant de résoudre le problème d'apprentissage sur des corpus de petite taille. Ceci nous a permis d'obtenir les mêmes résultats que l'état de l'art sur notre corpus, avec une méthode plus simple, ne nécessitant pas l'annotation de régions annotées sur les images.

Dans le chapitre 5, pour améliorer la représentation des images pour le calcul de similitude, nous avons proposé une méthode d'apprentissage des régions non supervisé. Elle permet, en maximisant l'entropie croisée sur l'image à différentes échelles, et sur différentes régions, d'apprendre les régions les plus susceptibles de contenir un objet d'intérêt. Ce qui nous a amené à développer une nouvelle fonction objectif, basé sur celle proposée précédemment, mais qui ajoute la classification de la région d'intérêt. Ceci nous permet d'avoir de meilleurs résultats que les solutions de l'état de l'art sur nos corpus, en passant de 92.73% à 94.55% en précision à un (plus proche voisin) et de 65.49% à 83.00% pour la MAP (Mean Average Precision).

Cette thèse propose aussi une solution au problème de la détection de gestes en mobilité. Nous avons proposé deux nouveaux blocs de convolutions, S1-FW et S2-FW, qui permettent de conserver les informations provenant de différentes trames de la vidéo. Cela nous a permis de proposer une nouvelle micro-architecture de réseau profond qui réalise une fusion tardive des informations temporelles. Nous avons étudié les effets de la position de la fusion des informations temporelles dans le réseau, et nous avons noté que la fusion tardive donne de meilleurs résultats. Pour l'utilisation sur mobile, nous préconisons une fusion précoce pour limiter le nombre de paramètres, avec un gain d'environ 20%. La micro-architecture que nous avons proposée obtient des résultats similaires aux approches de l'état de l'art sur notre corpus de reconnaissance de geste, mais avec un nombre réduit de paramètres.

Cependant, l'utilisation de micro-architecture pour la tâche de reconnaissance d'instances n'est pour le moment pas envisageable. L'écart de performance entre des réseaux type AlexNet et ResNet sont important, et nous préconisons l'utilisation de réseau plus profonds pour une meilleure représentation des images. Pour GUIMUTEIC, le choix le plus évident semble être de ne détecter les œuvres que lorsque que l'utilisateur réalise un geste ou la réponse nécessite une information sur l'environnement du visiteur.

Un ensemble de collections d'images et de vidéos ont été créées pour évaluer toutes les propositions faites dans cette thèse. Ces collections sont mises à disposition en accès libre à la communauté.

Les travaux présentés dans cette thèse peuvent donner lieu à de nombreux travaux futurs. Dans un premier temps, la flexibilité de la météo-architecture proposée pour la recherche d'instances permet son utilisation avec différents modèles de réseaux de neurones.

Des modèles plus récents comme Inception ou DenseNet pourraient améliorer les résultats. Ceci permettrait par la suite de réfléchir à une diminution du nombre de paramètres et de complexité du réseau, pour faciliter l'utilisation mobile. Pour la reconnaissance de gestes, la micro-architecture proposée a tendance à sur-apprendre plus facilement que les modèles de l'état de l'art. C'est un problème qu'il faut adresser pour envisager d'autres applications basées sur cette architecture.

Cette thèse soulève des problématiques à plus long terme pour l'aide à la visite de lieux touristiques. La fusion dans un seul réseau des deux problèmes que nous avons adressés, la recherche d'instances et la détection de gestes, est envisageable avec des approches d'apprentissage multi-tâches (multi-task learning). Cela permettrait une économie de temps de calcul et de mémoire considérable pour l'utilisation mobile. Le dispositif GUIMUTEIC est composé d'autres capteurs en plus de la caméra, avec par exemple des accéléromètres, un magnétomètre et un gyroscope. Des approches multi-modales pour l'apprentissage de l'environnement sont donc possibles. Pour aller plus loin, nous envisageons également l'apprentissage des parcours type dans le musée, pour une visite guidée basée sur les habitudes des visiteurs. Ceci permettrait d'envisager GUIMUTEIC comme un vrai guide, et non plus comme un simple assistant de visite, comme il a été demandé dans les études participatives. Nous espérons que la mise à disposition des collections présentées dans cette thèse pourra aider pour l'étude de ces problématiques.

A

Corpus pour l'évaluer des méthodes de recherches d'information multimédia

A.1 CORPUS D'IMAGES POUR LA RECHERCHE D'INSTANCE DANS UN MUSÉE D'ART

Le premier corpus que nous avons construit se nomme "CLICIDE". Il est composé de photographies de peintures issues de l'exposition permanente du musée de Grenoble¹. Ce musée propose au visiteur principalement des peintures occidentales entre le XIV^{ème} et le XXI^{ème} siècle. Il y a donc une grande variabilité de styles et d'époques (expressionnisme, impressionnisme, art sacré, pop art, ...). La figure A.1 présente quelques images montrant la diversité des œuvres de cette collection.

Le corpus Clicide est composé de 3425 photographies, qui représentent 473 œuvres du musée. Les œuvres ont été photographiées par 3 personnes, en utilisant un appareil reflex et des téléphones portables. Chaque œuvre est photographiée plusieurs fois (une image de l'œuvre complète, et des images correspondant à des parties de l'œuvre). Pour chaque œuvre considérée, une photographie du cartouche est également stockée. Chaque image d'œuvre est associée à un identifiant unique sous la forme suivante : <numéro de salle>_<numéro de l'œuvre dans la salle>_<numéro d'indice>. Ces images sont rognées manuellement afin de limiter la proportion d'arrière-plan.

De plus, 177 photographies, de 143 œuvres, tirées aléatoirement de la collection initiale, sont utilisées comme requêtes (et donc retirées du corpus). Ces images sont prises de différents points de vus et avec différentes proportions d'arrière-plan. La figure A.2 présente une image requête (à gauche) et une image du même objet tirée du corpus (à droite).

1. <http://www.museeegrenoble.fr/>



Figure A.1 : Images tirées de la collection Clicide. de gauche à droite : “Portrait de la mère du docteur Bordier” de Hippolyte Flandrin, “Les fruits” de Séraphine de Senlis, “O Combate” de Vicente do Rego Monteiro.

A.2 CORPUS DE RECHERCHE D’IMAGE ET DE VIDÉO DANS UN MUSÉE D’HÉRITAGE CULTUREL

Une deuxième collection a été réalisée avec le musée Gallo-Romain de Fourvière² qui est un musée français, localisé à Lyon, portant sur la civilisation Gallo-Romaine. Situé près d'un théâtre romain sur la colline de Fourvière, ce musée présente dans sa collection permanente, des objets pré-romains, romains, celtes (inscriptions, statues, joaillerie, objets de la vie courante), comme le montre les exemples de la figure A.3.

La collection appelée **GaRoFou**, pour musée **Gallo Romain de Fourvière**, est composée d'images fixes (**GaRoFou_I**) et de vidéos (**GaRoFou_V**).

A.3 GAROFOU_I

GaRoFou_I contient au total 1252 photographies, prises par des appareils reflex, de 311 œuvres. Parmi ces images, 1068 sont des images du corpus, et 184 des images requêtes sélectionnées aléatoirement. Sur les 311 œuvres, 166 sont représentées dans l'ensemble des requêtes. Les photographies requêtes ont différents points de vue, qui ne sont pas forcément présents dans le corpus. Chaque image du corpus est identifiée par l'œuvre qui y est visible, auquel est associé le *type* de l'œuvre parmi : stèle (roches gravées, colonnes, ...), statue (sculptures, reliefs, ...), poterie, et autres (pièces, joaillerie, ...). Les œuvres sont identifiées par un triplet numéro de niveau (de A à D), le numéro d'œuvre dans le niveau, et un numéro d'image de l'œuvre suivant le format suivant :<numéro de niveau>_<numéro de l'œuvre>_<numéro d'image de l'œuvre>. Les images d'une même œuvre sont donc identifiées spécifiquement.

2. <http://www.museegalloromain.grandlyon.com/>



(a) Exemple de requête



(b) Image référence du corpus

Figure A.2 : Photographies d'“Animaux Fleurs et Fruits” d'Alexandre-François Desportes tirée de Clicide.

A.4 GAROFOU_V

La partie vidéo de la collection Garofou est composée de 11 vidéos qui correspondent à des visites de différents étages du musée. Ces visites ont été effectuées par 5 personnes différentes, le même jour, avec une caméra fixée au dessus de la tête. Le détail sur ces vidéos est présenté dans le tableau A.1. Dans ce tableau, nous détaillons en particulier les étages (notés de A à D) du musée, la durée totale des vidéos brutes dans lesquelles les personnes ne sont pas forcément devant une œuvre, les durées durant lesquelles l'annotation manuelle a déterminé que des œuvres étaient le centre d'intérêt des visiteurs, le nombre d'œuvres correspondantes qui peut contenir des redondances car une personne peut se focaliser plusieurs fois sur une œuvre, ainsi que le nombre d'œuvres uniques vues. Associés à chaque vidéo, les objets visibles qui ont attiré l'attention du visiteur sont indiqués par leur horodatage d'apparition et de disparition (en hh :mm :ss par rapport au début de vidéo). Pour générer ces annotations, nous avons développé une interface spécifique sur la base de



Figure A.3 : Photographies de la collection de Fourvière. De gauche à droite : une stèle, une statue et une poterie.

la structure d'annotation du projet CAMOMILE [140], dont un exemple est présenté en figure A.4.

étage	# vidéos	durée totale	durée avec œuvre	# d'œuvres regardées	# œuvres uniques
A	4	66'55"	38'35"	157	59
B	2	31'21"	* 14'44"	77	56
C	3	30'51"	* 17'35"	63	37
D	5	57'34"	* 25'06"	115	63
total	11	186'41"	96'00"	412	215

Table A.1 : Vidéos de la collection GaRoFou_V.

De la partie annotée de cette collection sont tirées des images fixes : chaque segment (suite d'images consécutives temporellement) annoté a été découpé en, au plus, 10 sous-segments de 1 seconde répartis régulièrement et sans chevauchement. De chaque sous-segment, l'image la plus nette est sélectionnée par recherche de plus grande variance de couleurs après application d'un opérateur laplacien. Pour évaluer les systèmes, chacune de ces images extraites d'un visiteur est utilisée comme requête face aux images extraites issues des visites des autres visiteurs (validation croisée par visiteur). Nous conservons dans les requêtes uniquement les œuvres qui ont été vues par au moins deux visiteurs. Le tableau A.2 récapitule les données quantitatives liées aux images extraites de GaRoFou_V, par utilisateur : nous détaillons en particulier le nombre de segments utilisés pour extraire les images annotées qui servent à réaliser les évaluations.



Figure A.4 : Interface d'annotation des vidéos. En haut : l'affichage de la vidéo, en bas chaque segment annoté de vidéo avec l'identifiant d'objet.

visiteur	# vidéos	durée totale	durée avec œuvre	# segments annotés	# d'œuvres uniques	# Images total	# Images requêtes
u1	2	48'54"	22'22"	115	101	768	625
u2	1	24'25"	14'8"	62	51	493	444
u3	4	63'44"	24'36"	153	141	964	624
u4	1	10'50"	2'24"	13	11	101	101
u5	3	38'49"	16'1"	69	60	453	338
total	11	186'41"	79'30"	412'	215	2779	2132

Table A.2 : Images requêtes issues des vidéos du corpus GaRoFou

	Clicide	Garofou	
		Garofou_I	Garofou_V
objet	Peintures (2D)	2D et 3D	2D et 3D
média	Images fixes	Images fixes	vidéos
acquisition	contrôlée	contrôlée	contrôlée
taille (C/Q)	2500 / 512	1100 / 172	2779/2132

Table A.3 : Caractéristiques des collections proposées

A.5 ANALYSE DE L'INTERACTION À BASE DE GESTE DANS LE CADRE DE VISITES CULTURELLES

Dans cette section, nous proposons un corpus de vidéos de gestes de la main avec vue à la première personne). La production d'une collection de vidéos annotées est long et coûteux, cependant un grand nombre de données est généralement nécessaire pour l'apprentissage automatique. Pour faciliter la génération de nouveau contenu, une partie de ce corpus est réalisé devant un fond-vert.

A.5.1 PRISE DE VUE RÉELLE

Une partie de ce corpus a été réalisé à partir d'un ensemble de 5 personnes, équipé de caméra autour de cou, réalisant 5 gestes différents. La caméra est placée de manière à avoir une vue à la première personne, au niveau de la poitrine, pour une vision optimale des mains. Dans la figure A.5, nous pouvons voir les images de chacuns des gestes présents.

Toutes les vidéos ont été réalisées en intérieur, avec des personnes marchant ou s'arrêtant aléatoirement. Aucune consigne de positionnement n'ayant été donnée, pour permettre une grande disparité des gestes et de leur durée, des fonds, et des mouvement réalisés. Les 29 heures de vidéos générée représentent 1669 différents gestes, ainsi que 1830 segments de vidéos sans gestes avec des fond similaire, pouvant être utilisés pour détecter les moments sans gestes. La tableau A.4, récapitule le nombre d'image pour chacuns des gestes. Les segments de transition entre chaque geste contiennent généralement très peu d'images mais sont très nombreux, leur total étant de 23023 images.



Figure A.5 : Les 5 gestes à reconnaître

Table A.4

	#Images
Total	120 055
g1	8890
g2	8792
g3	9062
g4	9236
g5	8800
Sans main	52252
Transition	23023

Table A.5 : Nombres d'images pour chaque geste présent dans la collection

A.5.2 GESTES SUR FOND VERT

L'avantage de créer des vidéos de gestes devant un fond vert est le suivant : la segmentation de la main du bras à reconnaître dans l'image est entièrement automatisable ; il est alors possible d'intégrer ultérieurement autant d'arrière-plans différents de l'image acquise que souhaité.

La figure A.6 présente l'installation utilisée pour produire le corpus. Le matériel nécessaire (un fond vert sur la structure amovible, 3 lumières et 2 réflecteurs) est facilement disponible et peu coûteux.

Pour capturer une vidéo d'apprentissage des données, le processus est le suivant : une personne est équipée d'une caméra qui filme devant la personne (smartphone ou caméra positionné sur la poitrine). La personne est placée face à un fond vert où l'éclairage est contrôlé pour éviter les ombres sur l'écran (2 lumières de chaque côté dirigées vers des diffuseurs qui sont dirigés vers le fond vert, une lumière sur la personne orientée vers le plafond). La personne effectue la série de gestes requise dans un ordre spécifique.

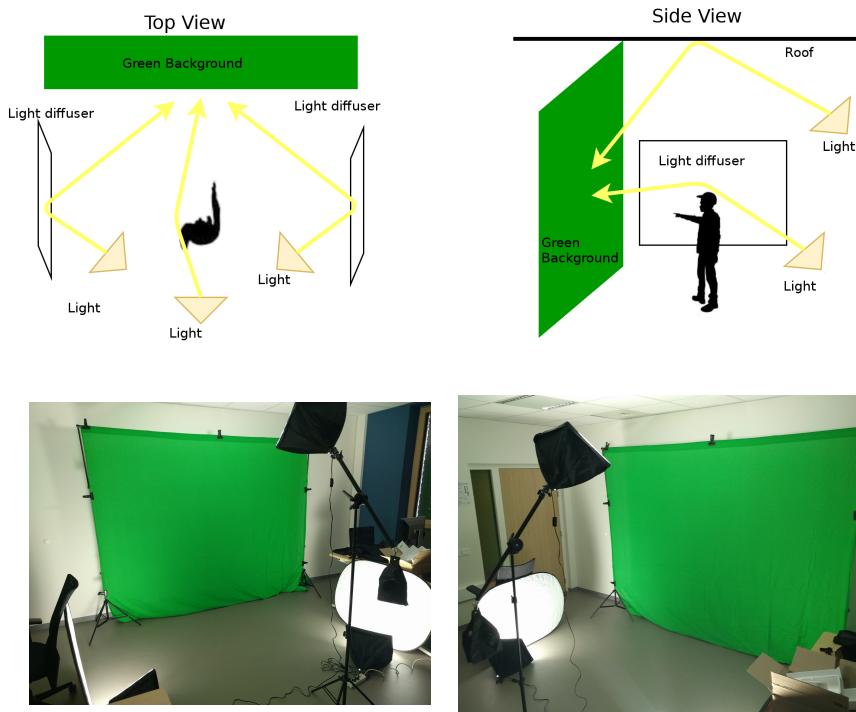


Figure A.6 : Installation du fond-vert (www.videomaker.com)

Dix personnes ont participé à la création de cette partie du corpus. Ils ont chacun fait 5 fois chaque geste avec les deux mains, ce qui correspond à 50 gestes par personne, 500 gestes au total. Dans la figure A.7, on peut voir que les participants n'ont pas effectué les gestes de la même manière. Pour plus de généralité, nous laissons les gens libres d'effectuer les gestes à leur manière.

A posteriori, les vidéos sont collectées et un algorithme supprime le fond vert. A partir du résultat, un masque est créé avec la main en blanc sur fond noir (voir figure A.8.) Ce masque correspond à la vérité de la segmentation spatio-temporelle du geste. fait pour vérifier / corriger que l'ordre a été respecté et que l'algorithme de détection de mouvement n'a pas échoué (moins de cinq minutes pour cinquante gestes capturés).

Cette solution, en plus d'obtenir une bonne segmentation automatique de la main en supprimant le fond vert, permet le changement de l'arrière-plan afin de générer de nouvelles données d'apprentissage à moindre coût tout en ayant déjà les annotations associées.

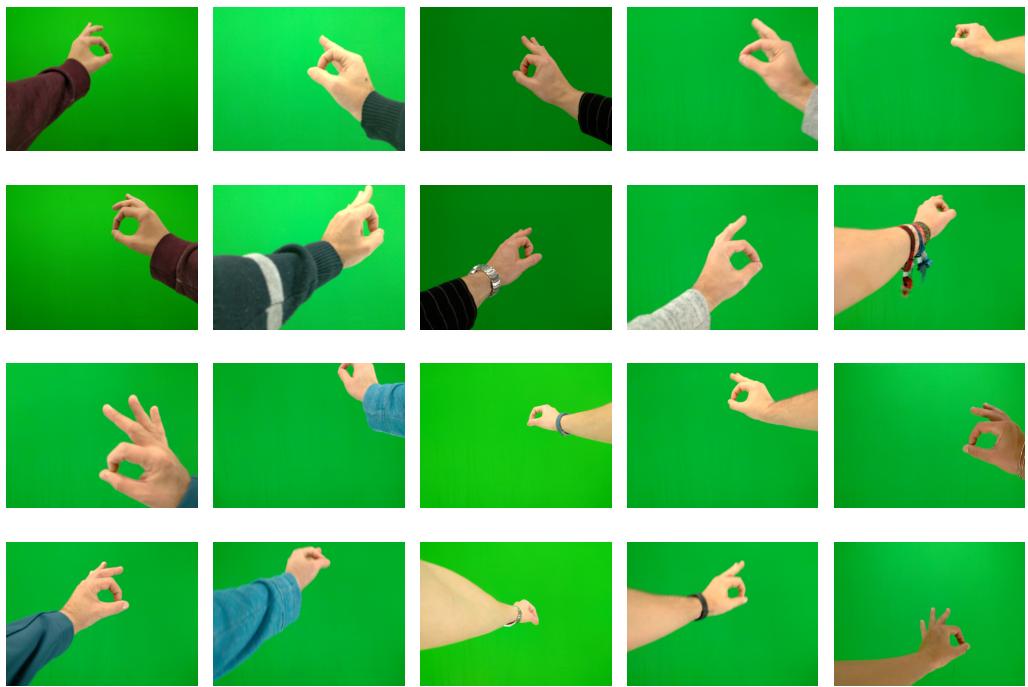


Figure A.7 : Le même geste fait par différentes personnes. Peu d'instructions ont été données pour l'apparence désirée du geste



Figure A.8 : Le geste capturé sur un fond vert, avec un arrière-plan intégré et la segmentation automatique

B

Etudes de la pertinence de l'interaction à base de geste

Des séances de conception participatives ont été réalisées avec l'entreprise Multicom¹. L'objectif de ces séances était de réunir des utilisateurs potentiels du dispositif GUIMUTEIC, et de les faire s'entretenir sur les applications éventuelles, et l'interaction possible avec l'appareil. Trois séances différentes ont été réalisées, avec chacune leurs objectifs séparés.

B.1 LE DISPOSITIF PROPOSÉ

Ces séances ont pour but de réfléchir aux possibilités du dispositif GUIMUTEIC. Ce qui est présenté aux participants est que l'appareil est constitué d'un casque audio, avec la possibilité d'un écran transparent, rabattable ou amovible. Cette écran peut être sous la forme de lunette, ou de petit écran de coin de l'oeil. Il peut également être équipé d'une surface tactile, sur l'écran ou sur un boîtier déporté.

Au niveau des applications, cette appareil est décrit comme une aide à la visite touristique, notamment dans le cadre des musées. Les participants sont libre d'imaginer comment ce guide peut être utilisé, et quel est son objectif.

Ce dispositif n'est pas fixe, et les participants vont être amené à réfléchir aux améliorations possible. Ce premier dispositif n'a pour but que de présenter une solution possible au projet GUIMUTEIC, aucune limite n'étant fixée aux participants quant à leurs idées.

1. <http://www.multicom-ergonomie.com/>

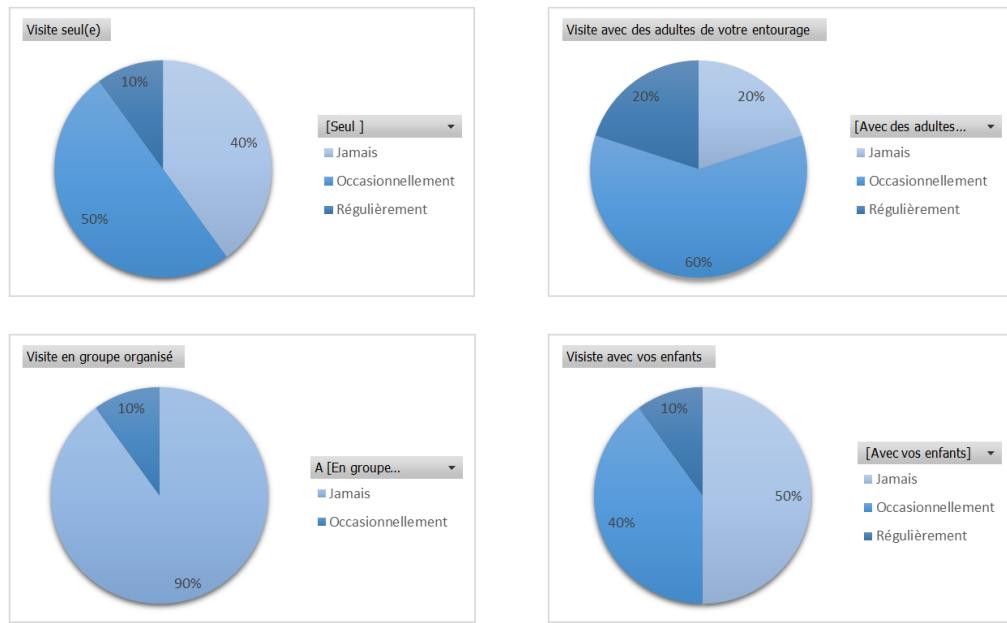


Figure B.1 : Répartition des groupes de personnes en fonction de leur habitude de visite.

B.2 LES PARTICIPANTS

Dix participants ont été réunis pour ces séances. Les profils ont été sélectionnés pour être varié, avec autant d’hommes que des femmes, des personnes actives, des étudiants ou des retraités, avec ou sans enfant. La figure B.1 présente la répartition des participants en fonction de leurs habitudes de visite. La majorité des participants n’ont jamais eu recours à un guide pour une visite organisée, et on voit une répartition équitable entre les personnes visitant les musées seuls ou en groupes.

B.3 SÉANCES PARTICIPATIVES

B.3.1 SÉANCE 1

La première séance avait pour but d’obtenir une vue générale des possibilités d’applications du dispositif GUIMUTEIC. Elle devait répondre aux objectifs suivants :

- Connaître les pratiques actuelles des participants lors de visite de musée/monument;
- Avoir la perception et le ressenti vis-à-vis du dispositif prévu;
- Définir des modes d’interaction souhaités;
- Définir des fonctionnalités innovantes souhaitées.

Suite à cette séance, les fonctionnalités préférées par les participants sont :

- L’analyse du regard pour proposer des informations correspondantes
- L’orientation du visiteur en fonction d’un parcours de visite



(a) Démarrer

(b) Sélectionner

Figure B.2 : Gestes Sélectionner et Démarrer identifiés lors des séances participatives

- La commande vocale
- Le contrôle à base de gestes

Les participants ont relevé deux utilisations distincts du dispositif GUIMUTEIC : un assistant de visite et un guide de visite. Un assistant intervient lors de la visite pour apporter des informations complémentaires sous forme de commentaire audio ou vidéo. Un guide oriente l'utilisateur en fonction d'un parcours prédefini ou personnalisé pour le visiteur en fonction de son temps, ses connaissances, etc.

Un point important soulevé est que les participants privilégient l'utilisation de l'appareil de manière autonome. Ils ne souhaitent pas un objet supplémentaire pour l'utiliser.

Un certain nombre de souhaits sur le dispositifs ont été émis :

- Avoir un seul écouteur ou un casque ouvert pour pallier au sentiment d'isolement;
- Pouvoir orienter la caméra vers l'oeuvre ;
- Pour relever l'écran quand l'utilisateur ne souhaite pas l'utiliser.

B.3.2 SÉANCE 2

L'objectif de la deuxième séance était de déterminer l'intérêt d'un écran sur le dispositif, ainsi que l'ergonomie du casque, avec les buts suivants :

- Connaître le ressenti des participants vis-à-vis du dispositif.
- Définir l'interaction lors du parcours de visite.
- Identifier les gestes pertinents.

La présence d'un écran est jugé inutile si l'écran est trop petit. Le casque présenté avait un écran du même type de les GoogleGlass², mais rabattable. Les participants l'ont jugé trop petit pour être utilisable pendant une longue visite. De plus, la branche utilisée pour rabattre l'écran est perçue comme trop imposante et non adaptée aux enfants.

Un certain nombre de gestes sont proposés par les participants pour réaliser l'interaction. Les gestes suivants, qui n'ont pas été retenus car ils requièrent un certain niveau d'apprentissage : sauvegarder et reprendre. Deux autres gestes n'ont pas été retenus car ils sont similaires (voir figure B.2) : sélectionner et démarrer.

2. www.google.com

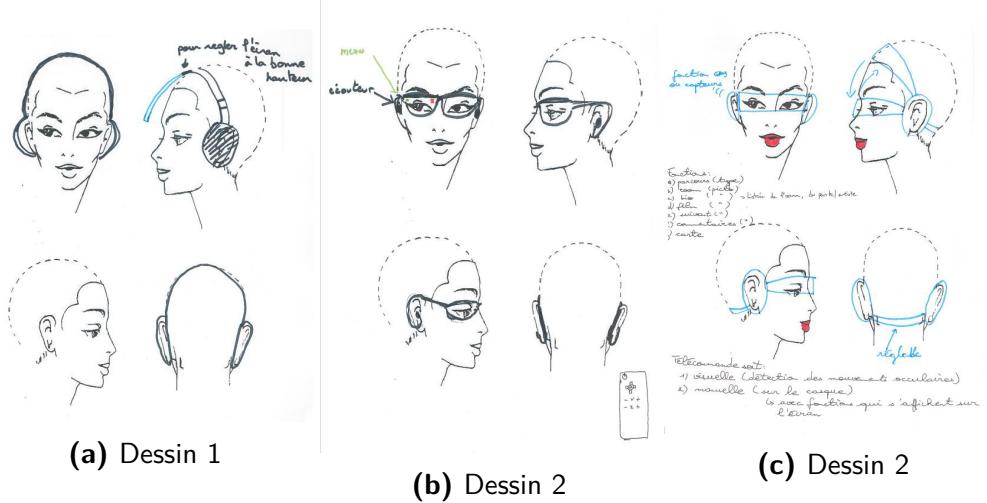


Figure B.3 : Gestes Sélectionner et Démarrer identifiés lors des séances participatives

Il ressort de cette séance que les participants sont enclins à utiliser des gestes, si ceux-ci sont “intuitifs et universels”. Ils ne souhaitent pas apprendre de nouveaux gestes. Certains avancent le fait qu’utiliser des gestes peut permettre de rendre l’interaction plus fluide et rapide. En revanche, ils insistent sur la rapidité et la qualité de la reconnaissance, qui doit être immédiate sous peine d’être rejetée par les utilisateurs. Le geste le plus demandé est le pointage, pour déclencher le guide sur une oeuvre donnée.

B.3.3 SÉANCE 3

La troisième séance devait répondre aux objectifs suivants :

- Définir l’ergonomie du casque (écouteur, écran, ...);
- Identifier les gestes et les modes d’interactions pertinents

Durant cette séance, les participants ont été invités à dessiner l’appareil idéal pour chacun. Les dessins sont ensuite soumis aux votes. Sur la figure B.3, on voit les dessins qui ont reçu le plus de votes. Le premier propose un écran et des écouteurs amovibles, avec une zone tactile au niveau des écouteurs pour le contrôle de l’appareil. La deuxième proposition est un système à base de lunettes, avec des écouteurs amovibles. L’interaction se fait alors par la reconnaissance des mouvements oculaires ou avec une télécommande. Le troisième dessin propose un écran semi-opaque et des écouteurs amovibles. L’interaction se fait également avec reconnaissance de mouvements oculaires.

Le bilan de cette séance est que le casque doit être réglable, et surtout amovible, pour ne pas couper l’utilisateur de reste du monde. Les écouteurs doivent être amovibles ou ouverts, et l’interaction doit être fait sans appareil supplémentaire.

B.4 BILAN

D nombreux éléments pour le dispositif GUIMUTEIC peuvent être retenus de ces séances de conceptions participatives. Tout d’abord au niveau du guide, un choix entre une aide

forte, avec guidage du visiteur, et une aide simple, qui n'intervient que lorsque qu'on lui demande, est important. Ensuite, au niveau du dispositif, la flexibilité de son utilisation est un point soulevé par de nombreux participants. Les écouteurs doivent être amovibles ou ouverts, pour que le visiteur puisse communiquer avec d'autres personnes s'il le désire. Si un écran est disponible, il doit être rabattable, pour ne pas gêner l'utilisateur.

Au niveau de l'interaction avec l'appareil, un consensus sur le fait qu'il ne faut pas manipuler l'appareil supplémentaire est apparu. L'analyse de mouvement oculaire est revenue plusieurs fois, mais n'est pas compatible avec un écran rabattable, en plus d'être difficile à mettre en place pour un appareil destiné au grand public. L'interaction à base de commande vocale est bien accueilli, mais difficilement utilisable dans un musée. Les gestes sont donc ce qu'il faut retenir dans ce cas pour une interaction adaptée, si la reconnaissance est assez rapide et précise.

C

Réseau de reconnaissance de gestes

Cette section présente le code et les schémas des blocs et du réseau présentés dans la chapitre 6. Le code est en python, pour le framework PyTorch, et permet de voir les détails de l'implémentation de chacun des blocs. Les schémas sont des compléments à la figure 6.7, qui ne présentait qu'un des exemples d'architecture utilisé pour la courbe 6.8. Nous fournissons également les schémas des architecture ShuffleNet et ResNet modifier pour réaliser la fusion des informations, présentés dans le tableau 6.7.

C.1 IMPLÉMENTATION

C.1.1 BLOC S1-FRAMEWISE

```
1 class S1FW(nn.Module):
2     """
3         S1 FrameWise module
4         inputs :
5             * inp = number of input channel
6             * outp = number of output channel
7             * nbf = number of frames
8     """
9
10    def __init__(self, inp, outp, nbf=1):
11        super(S1FW, self).__init__()
12        # deepwise 3*3 convolution
13        self.conv3 = nn.Conv2d(inp*nbf, inp*nbf,
14                            kernel_size=3, stride=1,
15                            padding=1, groups=inp*nbf)
16        self.bn3 = nn.BatchNorm2d(inp*nbf)
17        # framewise 1*1 convolution
18        self.conv1 = nn.Conv2d(inp*nbf, outp*nbf,
19                            kernel_size=1, stride=1,
```

```

21                                     padding=o, groups=nbf)
22         self.bn1 = nn.BatchNorm2d(outp*nbf)
23         self.relu = nn.ReLU(True)
24
25     def forward(self, x):
26         right = self.relu(self.bn3(self.conv3(x)))
27         right = self.relu(self.bn1(self.conv1(right)))
28         return x.add(right)

```

C.1.2 BLOC S2-FRAMEWISE

```

1  class S2FW(nn.Module):
2      """
3          S2 FrameWise module
4          inputs :
5              * inp = number of input channel
6              * outp = number of output channel
7              * nbf = number of frames
8      """
9
10     def __init__(self, inp, outp, nbf=1):
11         super(S2FW, self).__init__()
12         # deepwise 3*3 convolution
13         self.conv3      = nn.Conv2d(inp*nbf, inp*nbf,
14                               kernel_size=3, stride=2,
15                               padding=1, groups=inp*nbf)
16         self.bn3       = nn.BatchNorm2d(inp*nbf)
17         # framewise 1*1 convolution
18         self.conv1      = nn.Conv2d(inp*nbf, outp*nbf, kernel_size
19                               =1, stride=1, padding=o, groups=nbf)
20         self.bn1       = nn.BatchNorm2d(outp*nbf)
21         self.relu      = nn.ReLU()
22         self.pool      = nn.AvgPool2d(kernel_size=3, padding=1,
23                               stride=2)
24
25     def forward(self, x):
26         left = self.pool(x)
27         right = self.relu(self.bn3(self.conv3(x)))
28         if self.shuffle:
29             right = channel_shuffle(right, self.nb_groups)
30         right = self.relu(self.bn1(self.conv1(right)))
31         return torch.cat((left, right), 1)

```

C.1.3 RÉSEAU MULTI-TRAME

```

1  class DenseMobile(nn.Module):
2      def __init__(self, in_channel=3, nb_frames=3, num_classes=6):
3          super(DenseMobile, self).__init__()

```

```

5     self.blocStart = DownBloc(in_channel, 32-in_channel,
6     in_channel, nb_frames=nb_frames) # 224x224x3 -> 112x112x32
7     self.features = nn.Sequential(
8         S1FW(32,32,nb_frames), # -> 112
9         x112x32xnb_frame
10        S2FW(32,32,nb_frames), # -> 56
11        S1FW(64,64,nb_frames), # -> 56
12        x56x64xnb_frame
13        S2FW(64, 192,nb_frames), # -> 28
14        x56x128xnb_frame
15        FusionBloc(256,256, nb_frames),-> 28
16        x28x256
17        S1(256,256, 256), # -> 28x28x256
18        S2(256, 256, 256), # -> 14x14x512
19        S1(512, 512, 512), # -> 28x28x256
20        S2(512,512,512), # -> 7x7x1024
21        nn.AvgPool2d(kernel_size=7, padding=0,
22        stride=1), # 3x3
23    )
24
25     self.classif = nn.Linear(1024, num_classes)
26
27 def forward(self, x):
28     x = self.blocStart(x)
29     x = self.features(x)
30     return self.classif(x.view(x.size(0),-1))

```

C.2 FUSION DES INFORMATIONS

C.2.1 RÉSEAU MULTI-FRAME

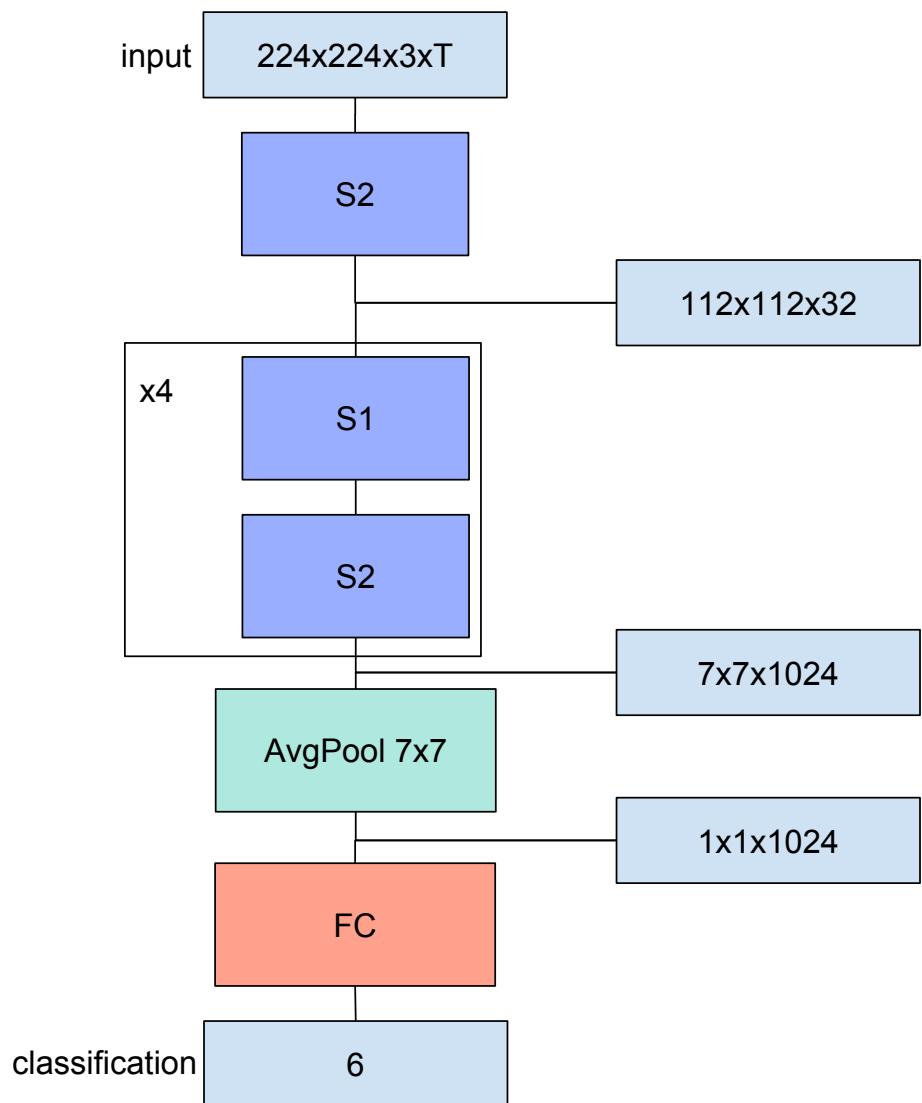


Figure C.1 : Fusion 1

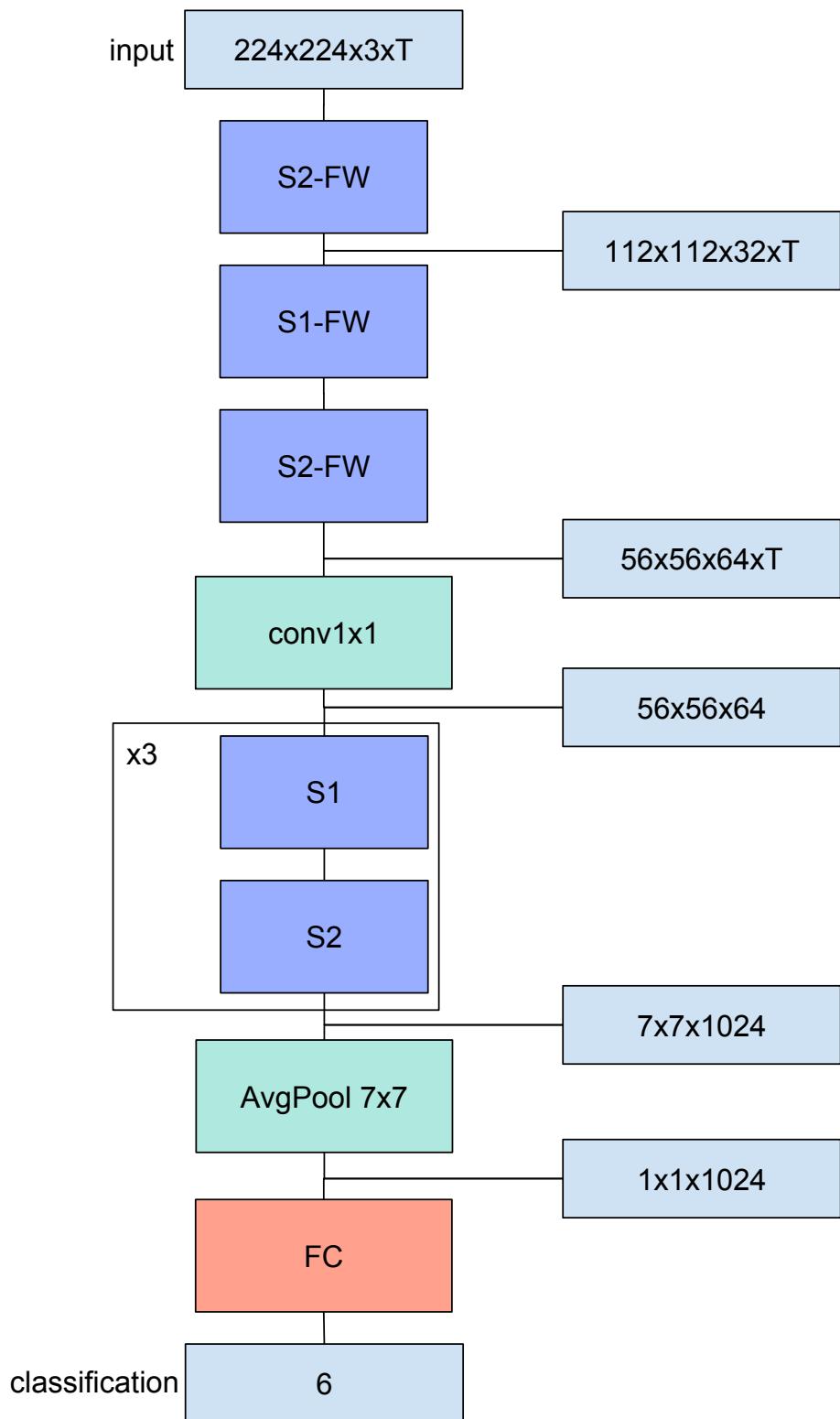


Figure C.2 : Fusion 2

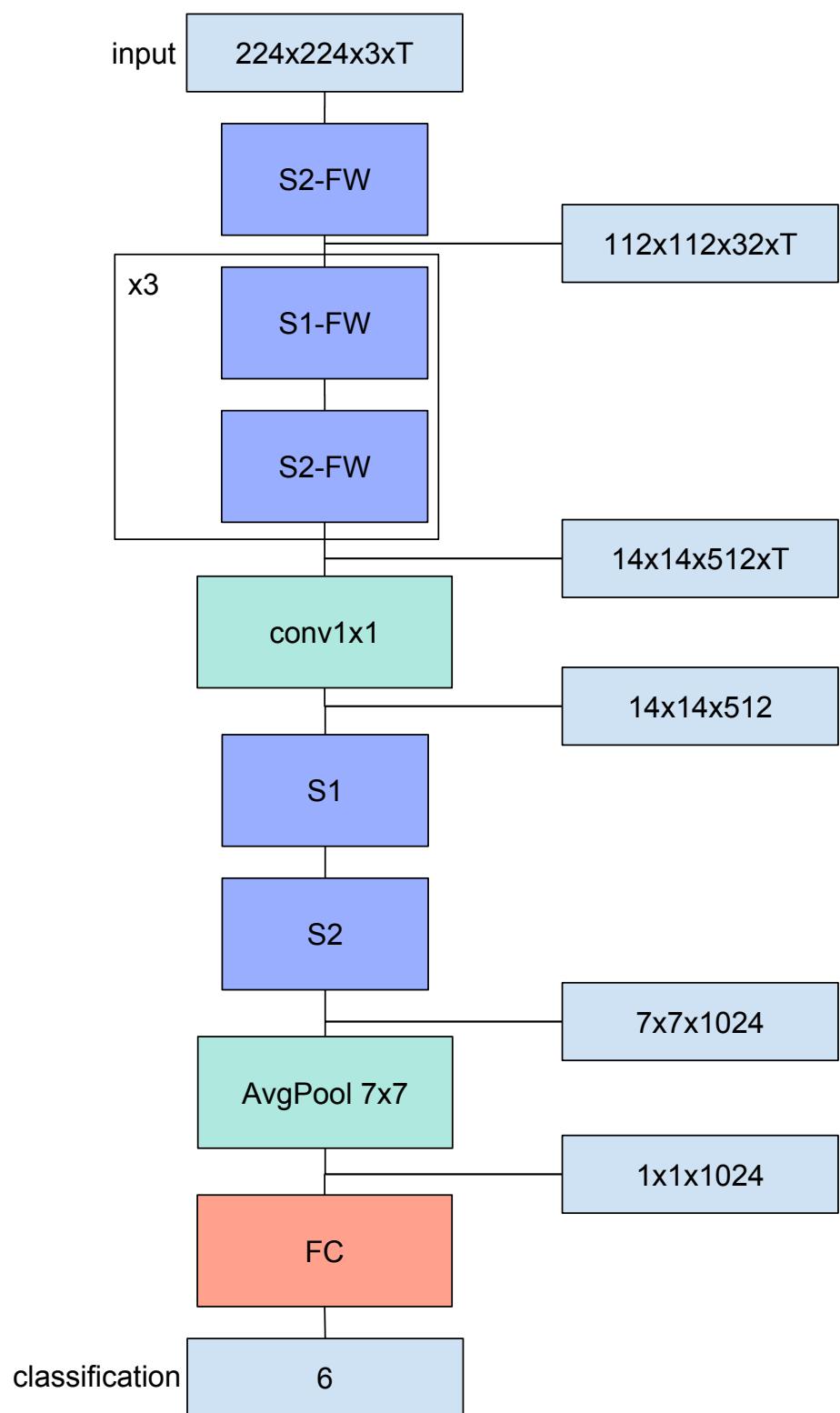


Figure C.3 : Fusion 4

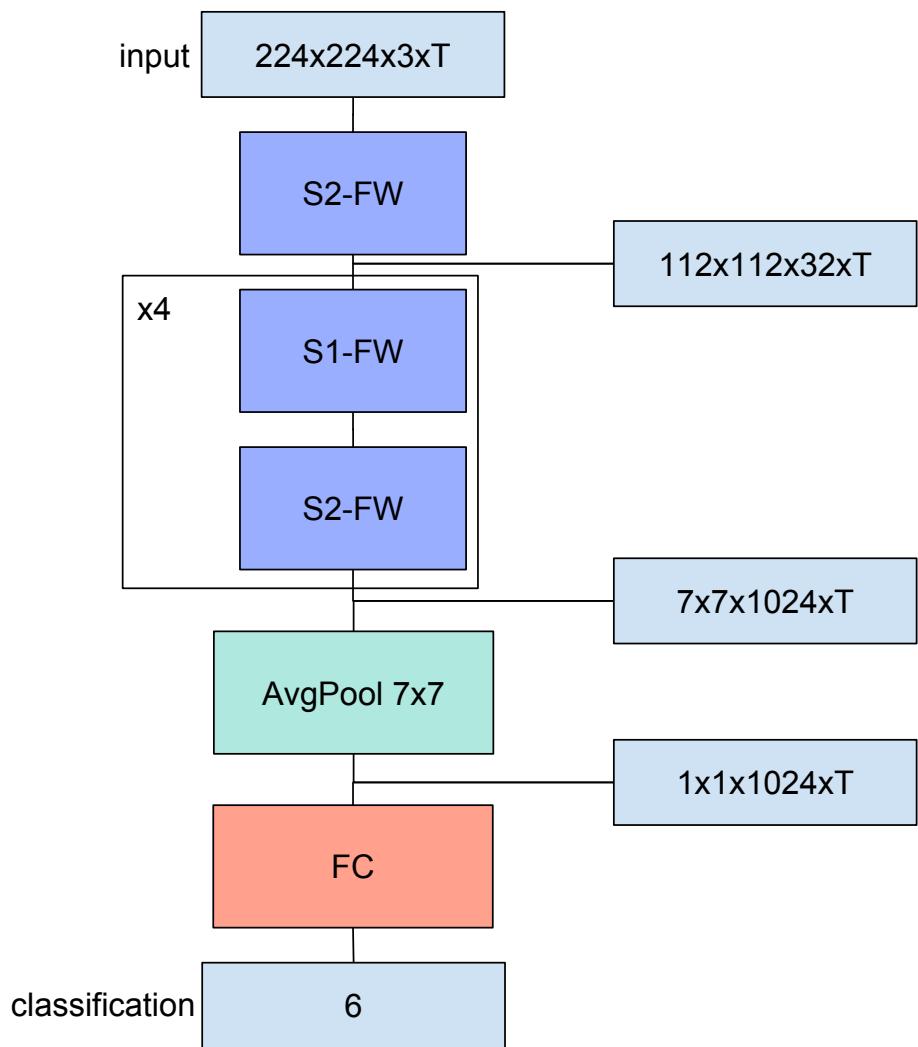


Figure C.4 : Fusion 5

Bibliographie

- [1] Lorenzo Baraldi, Francesco Paci, Giuseppe Serra, Luca Benini, and Rita Cucchiara. Gesture recognition using wearable vision sensors to enhance visitors' museum experiences. *IEEE Sensors Journal*, 15(5) :2705–2714, 2015.
- [2] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1) :1929–1958, 2014.
- [3] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.
- [4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014.
- [6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers : Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [8] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. SqueezeNet : Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size. *arXiv preprint arXiv :1602.07360*, 2016.
- [9] Y Ioannou, D Robertson, R Cipolla, and A Criminisi. Deep roots : Improving cnn efficiency with hierarchical filter groups. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*. IEEE, 2017.
- [10] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet : An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv :1707.01083*, 2017.

- [11] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [12] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.
- [13] Liang Zheng, Yi Yang, and Qi Tian. Sift meets cnn : A decade survey of instance retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 40(5) :1224–1244, 2018.
- [14] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a “siamese” time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.
- [15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet : A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.
- [16] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.
- [17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553) :436, 2015.
- [18] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.
- [19] Stephen Bitgood. When is “museum fatigue” not fatigue? *Curator : The Museum Journal*, 52(2) :193–202, 2009.
- [20] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets : Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [21] Joel Lanir, Tsvi Kuflik, Eyal Dim, Alan J Wecker, and Oliviero Stock. The influence of a location-aware mobile guide on museum visitors’ behavior. *Interacting with Computers*, 25(6) :443–460, 2013.
- [22] Rebecca E Grinter, Paul M Aoki, Margaret H Szymanski, James D Thornton, Allison Woodruff, and Amy Hurst. Revisiting the visit :: understanding how technology can shape the museum visit. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 146–155. ACM, 2002.

- [23] Ben Gammon and Alexandra Burch. Designing mobile digital experiences. *Digital technologies and the museum experience : Handheld guides and other media*, 35, 2008.
- [24] Florence Andreacola. Musée et numérique, enjeux et mutations. *Revue française des sciences de l'information et de la communication*, 5, 2014.
- [25] Albrecht Schmidt, Michael Beigl, and Hans-W Gellersen. There is more to context than location. *Computers & Graphics*, 23(6) :893–901, 1999.
- [26] Maxime Portaz, Matthias Kohl, Jean-Pierre Chevallet, Georges Quénnot, and Philippe Mulhem. Object instance identification with fully convolutional networks. *Multimedia Tools and Applications*, pages 1–18, 2018.
- [27] Maxime Portaz, Matthias Kohl, Georges Quénnot, and Jean-Pierre Chevallet. Fully convolutional network and region proposal for instance identification with egocentric vision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2383–2391. IEEE, 2017.
- [28] Maxime Portaz, Johann Poignant, Mateusz Budnik, Philippe Mulhem, Jean-Pierre Chevallet, and Lorraine Goeuriot. Construction et évaluation d'un corpus pour la recherche d'instances d'images muséales. In *CORIA*, pages 17–34, 2017.
- [29] Maxime Portaz, Philippe Mulhem, and Jean-Pierre Chevallet. Étude préliminaire à la recherche de photographies muséales en mobilité. In *CORIA 2016 COnférence en Recherche d'Information et Applications 2016*, pages 335–344, 2016.
- [30] Maxime Portaz, Mateusz Budnik, Philippe Mulhem, and Johann Poignant. Mrim-lig at imageclef 2016 scalable concept image annotation task. In *CLEF (Working Notes)*, pages 363–370, 2016.
- [31] Simon Knell. The shape of things to come : museums in the technological landscape. *Museums in a digital age*, 1(3) :435, 2010.
- [32] Paul F Marty and Katherine Burton Jones. *Museum informatics : People, information, and technology in museums*, volume 2. Taylor & Francis, 2008.
- [33] Paul F Marty. My lost museum : User expectations and motivations for creating personal digital collections on museum websites. *Library & information science research*, 33(3) :211–219, 2011.
- [34] Eileen Hooper Greenhill. *Museums and the Shaping of Knowledge*. Routledge, 1992.
- [35] Suzanne Keene. Becoming digital. *Museum Management and Curatorship*, 15(3) :299–313, 1996.
- [36] John H Falk. *Identity and the museum visitor experience*. Routledge, 2016.

- [37] Tsvi Kuflik, Oliviero Stock, Massimo Zancanaro, Ariel Gorfinkel, Sadek Jbara, Sharar Kats, Julia Sheidin, and Nadav Kashtan. A visitor's guide in an active museum : Presentations, communications, and reflection. *Journal on Computing and Cultural Heritage (JOCCH)*, 3(3) :11, 2011.
- [38] James A Evans and Pat Sterry. Portable computers & interactive multimedia : a new paradigm for interpreting museum collections. *Archives and Museum Informatics*, 13(2) :113–126, 1999.
- [39] Allison Woodruff, Paul M Aoki, Rebecca E Grinter, Amy Hurst, Margaret H Szymanski, and James D Thornton. Eavesdropping on electronic guidebooks : Observing learning resources in shared listening environments. *arXiv preprint cs/0205054*, 2002.
- [40] S Angliss. Talking sense. *Museum Practice Magazine*, 34 :46–47, 2006.
- [41] Daniela Petrelli and Elena Not. User-centred design of flexible hypermedia for a mobile guide : Reflections on the hyperaudio experience. *User Modeling and User-Adapted Interaction*, 15(3-4) :303–338, 2005.
- [42] Palmyre Pierroux, Ingeborg Krane, and Idunn Sem. Bridging contexts and interpretations : Mobile blogging on art museum field trips. *MedieKultur : Journal of media and communication research*, 27(50) :18, 2011.
- [43] Alexandra Weilenmann, Thomas Hillman, and Beata Jungselius. Instagram at the museum : communicating the museum experience through social photo sharing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1843–1852. ACM, 2013.
- [44] Alar Kuusik, Sylvain Roche, and Frédéric Weis. Smartmuseum : Cultural content recommendation system for mobile users. In *Computer Sciences and Convergence Information Technology, 2009. ICCIT'09. Fourth International Conference on*, pages 477–482. IEEE, 2009.
- [45] Flavia Sparacino. The museum wearable : Real-time sensor-driven understanding of visitors' interests for personalized visually-augmented museum experiences. *Museums and the Web*, 2002.
- [46] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1) :1–127, 2009.
- [47] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4) :541–551, 1989.
- [48] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017.

- [49] Joseph Redmon and Ali Farhadi. Yolov3 : An incremental improvement. *arXiv*, 2018.
- [50] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn : Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [51] Keiichi Osako, Rita Singh, and Bhiksha Raj. Complex recurrent neural networks for denoising speech signals. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2015 IEEE Workshop on*, pages 1–5. IEEE, 2015.
- [52] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2 : End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182, 2016.
- [53] Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. Semeval-2016 task 4 : Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18, 2016.
- [54] Rajat Raina, Anand Madhavan, and Andrew Y Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880. ACM, 2009.
- [55] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.
- [56] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- [57] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2) :251–257, 1991.
- [58] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks : Tricks of the trade*, pages 437–478. Springer, 2012.
- [59] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv :1804.07612*, 2018.
- [60] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1) :3–55, 2001.
- [61] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016.
- [62] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks : Tricks of the trade*, pages 9–48. Springer, 2012.

- [63] Stephen José Hanson and Lorien Y Pratt. Comparing biases for minimal network construction with back-propagation. In *Advances in neural information processing systems*, pages 177–185, 1989.
- [64] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.
- [65] Sergey Ioffe and Christian Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
- [66] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.
- [67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [68] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 7, 2017.
- [69] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.
- [70] Jorge Sánchez and Florent Perronnin. High-dimensional signature compression for large-scale image classification. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1665–1672. IEEE, 2011.
- [71] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.
- [72] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017.
- [73] Babak Hassibi and David G Stork. Second order derivatives for network pruning : Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171, 1993.
- [74] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.
- [75] Song Han, Huizi Mao, and William J Dally. Deep compression : Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

- [76] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294, 2015.
- [77] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2) :129–137, 1982.
- [78] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net : Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.
- [79] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco : Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [80] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [81] Quoc V Le. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595–8598. IEEE, 2013.
- [82] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.
- [83] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.
- [84] Andrej Mikulik, Michal Perdoch, Ondřej Chum, and Jiří Matas. Learning vocabularies over a fine quantization. *International journal of computer vision*, 103(1) :163–175, 2013.
- [85] Giorgos Tolias and Hervé Jégou. Visual query expansion with or without geometry : refining local descriptors by feature aggregation. *Pattern recognition*, 47(10) :3466–3476, 2014.
- [86] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval : Learning global representations for image search. In *European Conference on Computer Vision*, pages 241–257. Springer, 2016.
- [87] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.
- [88] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2) :91–110, 2004.

- [89] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Simultaneous object recognition and segmentation by image exploration. In *European Conference on Computer Vision*, pages 40–54. Springer, 2004.
- [90] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10) :1615–1630, 2005.
- [91] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. Image matching using sift, surf, brief and orb : performance comparison for distorted images. *arXiv preprint arXiv:1710.02726*, 2017.
- [92] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 525–531. IEEE, 2001.
- [93] Luo Juan and Oubong Gwun. A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4) :143–152, 2009.
- [94] Liang-Chi Chiu, Tian-Sheuan Chang, Jiun-Yen Chen, Nelson Yen-Chung Chang, et al. Fast sift design for real-time visual feature extraction. *IEEE Transactions on Image Processing*, 22(8) :3158–3167, 2013.
- [95] Yan Ke and Rahul Sukthankar. Pca-sift : A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2004.
- [96] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3) :346–359, 2008.
- [97] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief : Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.
- [98] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb : An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.
- [99] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.
- [100] Josef Sivic, Andrew Zisserman, et al. Video google : A text retrieval approach to object matching in videos. In *iccv*, volume 2, pages 1470–1477, 2003.
- [101] Yimeng Zhang, Zhaoyin Jia, and Tsuhan Chen. Image retrieval with geometry-preserving visual phrases. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 809–816. IEEE, 2011.

- [102] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 2161–2168. Ieee, 2006.
- [103] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on*, pages 1–8. IEEE, 2007.
- [104] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European conference on computer vision*, pages 304–317. Springer, 2008.
- [105] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. Large-scale image retrieval with compressed fisher vectors. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3384–3391. IEEE, 2010.
- [106] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE, 2010.
- [107] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. Total recall : Automatic query expansion with a generative feature model for object retrieval. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [108] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2911–2918. IEEE, 2012.
- [109] Michal Perd’och, Ondrej Chum, and Jiri Matas. Efficient representation of local geometry for large scale object retrieval. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 9–16. IEEE, 2009.
- [110] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf : an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.
- [111] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.
- [112] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf : A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

- [113] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. From generic to specific deep representations for visual recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 36–45, 2015.
- [114] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *Proceedings of the IEEE international conference on computer vision*, pages 1269–1277, 2015.
- [115] Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. A baseline for visual instance retrieval with deep convolutional networks. In *International Conference on Learning Representations, May 7-9, 2015, San Diego, CA*. ICLR, 2015.
- [116] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *European Conference on Computer Vision*, pages 685–701. Springer, 2016.
- [117] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. *arXiv preprint arXiv:1511.05879*, 2015.
- [118] Florent Perronnin and Diane Larlus. Fisher vectors meet neural networks : A hybrid classification architecture. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3743–3752, 2015.
- [119] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *European conference on computer vision*, pages 392–407. Springer, 2014.
- [120] Mattis Paulin, Matthijs Douze, Zaid Harchaoui, Julien Mairal, Florent Perronnin, and Cordelia Schmid. Local convolutional features with unsupervised training for image retrieval. In *Proceedings of the IEEE international conference on computer vision*, pages 91–99, 2015.
- [121] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *European conference on computer vision*, pages 584–599. Springer, 2014.
- [122] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Cnn image retrieval learns from bow : Unsupervised fine-tuning with hard examples. In *European conference on computer vision*, pages 3–20. Springer, 2016.
- [123] Pierre Baldi and Yves Chauvin. Neural networks for fingerprint recognition. *Neural Computation*, 5(3) :402–418, 1993.
- [124] Jingen Liu, Jiebo Luo, and Mubarak Shah. Recognizing realistic actions from videos “in the wild”. In *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on*, pages 1996–2003. IEEE, 2009.

- [125] Heng Wang, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid. Evaluation of local spatio-temporal features for action recognition. In *BMVC 2009-British Machine Vision Conference*, pages 124–1. BMVA Press, 2009.
- [126] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pages 65–72. IEEE, 2005.
- [127] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011.
- [128] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
- [129] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011.
- [130] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.
- [131] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02) :107–116, 1998.
- [132] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997.
- [133] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Félix Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [134] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets : Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015.
- [135] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015.

- [136] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. Video description generation incorporating spatio-temporal features and a soft-attention mechanism. *arXiv preprint arXiv:1502.08029*, 2015.
- [137] Hervé Jégou and Ondřej Chum. Negative evidences and co-occurrences in image retrieval : The benefit of pca and whitening. In *Computer Vision–ECCV 2012*, pages 774–787. Springer, 2012.
- [138] Panu Turcot and David G. Lowe. Better matching with fewer features : The selection of useful features in large database recognition problems. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 2109–2116. IEEE, 2009.
- [139] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2 : Practical guidelines for efficient cnn architecture design. *arXiv preprint arXiv:1807.11164*, 2018.
- [140] Johann Poignant, Mateusz Budnik, Hervé Bredin, Claude Barras, Mickael Stefas, Pierrick Bruneau, Gilles Adda, Laurent Besacier, Hazim Ekenel, Gil Francopoulo, et al. The camomile collaborative annotation platform for multi-modal, multi-lingual and multi-media documents. In *LREC 2016 Conference*, 2016.