Machine learning and convolutional networks for an augmented expertise in biological dosimetry

Antonin Deschemps

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Université de Rennes

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Signal, Image, Vision*

Par

Antonin DESCHEMPS

Apprentissage machine et réseaux de convolution pour une expertise augmentée en dosimétrie biologique

Thèse présentée et soutenue à Rennes, le 19 décembre 2023
Unité de recherche : Centre Inria de l'université de Rennes

Rapporteurs avant soutenance :

Daniel RACOCEANU Professeur, Sorbonne Université
David ROUSSEAU Professeur, Université d'Angers

Composition du Jury :

Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour s'assurer qu'elle est conforme et devra être répercutée sur la couverture de thèse

Président :	Prénom NOM	Fonction et établissement d'exercice (*à préciser après la soutenance*)
Rapporteurs :	Daniel RACOCEANU	Professeur, Sorbonne Université
	David ROUSSEAU	Professeur, Université d'Angers
Examinatrices :	Maria VAKALOPOULOU	Maitre de conférence, Centrale Supélec
	Ruth WILKINS	Adjunct Research Professor, Carleton University
	Elisa FROMONT	Professeure, Université de Rennes
Dir. de thèse :	Charles KERVRANN	Directeur de recherche, Inria Rennes
Co-dir. de thèse :	Mohamedamine BENADJAOUD	Chercheur, IRSN / SERAMED

Table of Contents

List of Figures

NOTATIONS AND ACRONYMS

Notations

General

— $\mathbb{R}$: set of real numbers
— 1_K: vector where every component has value 1 of size K
— x^T: transpose of x
— $\text{diag}(x)$: square matrix with diagonal x
— $\|x\|_2$: L_2-norm of x
— θ: parameters of a model
— $\mathbb{E}$: expectation of random variable
— $\mathbb{KL}(X|Y)$: Kullback-Leibler divergence between distributions X and Y
— ∇_θ: gradient with respect to variable θ (usually model parameters)
— $\mathcal{H}_\theta$ Hessian with respect to θ
— $\mathcal{D}$: training dataset
— $\mathcal{J}$: batch of training data (random subset of $\mathcal{D}$)
— $|A|$: number of elements in set A, for example $|\mathcal{J}|$ is the batch size
— σ: activation function of a neural network
— $\frac{\partial f}{\partial x}$: partial derivative of f with respect to variable x
— $\mathcal{L}$: Risk criterion
— H, W: Height and Width of an image

Unet ensembling

— $\mathcal{R}$: Sparse Variation regularization term
— ρ: Sparse Variation sparsity parameter
— λ: regularization strength parameter
— M: number of models in ensemble
— $(u, v) \in \Omega$: pixel coordinates (Ω) is the set of all pixel coordinates, with $|\Omega| = H \times W$
— ϕ_θ: Unet model with parameters θ

— y_L: image downsampled by a factor L, i.e of size $H/L, W/L$
— $\nabla_{u,v}$: intensity gradients of image (with respect to image axes)
— $\mathbf{\Lambda}_{\theta_k}(x_i)$: binarized prediction of ϕ_{θ_k} on image x_i
— $\mathbf{\Lambda}_{GT}(y_i)$: binarized ground truth
— $\boldsymbol{I}_i$: intersection between binarized prediction and ground truth
— T_A: agreement threshold
— T_C: single-model confidence threshold
— $\boldsymbol{S}_i(u, v)$: sum of the binary predictions of ensemble members at location u, v for image x_i
— $\boldsymbol{D}_i(u, v)$: binary agregated decision at location u, v for image x_i

Acronyms

— IAEA: International Atomic Energy Agency / Agence Internationale pour l'Energie Atomique
— DC: Dicentric Chromosome
— MC: Monocentric Chromosome
— LRAcc: Laboratory for Radiobiology of Accidental Exposures / Laboratoire pour la Radiobiologie des Expositions Accidentelles
— IRSN: Radioprotection and Nuclear Safety Institute / Institut pour la Radioprotection et la Sûreté Nucléaire
— SERPICO: Space-TimE RePresentations, Imaging and cellular dynamics of Molecular COmplexes
— Inria: National Institute for Research in Informatics and Automatics / Institut National pour la Recherche en Informatique et Automatique
— SSL: Self-Supervised Learning
— ADCI: Automated Dicentric Chromosome Identififer
— ADS: Automated Dicentric Scoring
— DL: Deep Learning
— CNN: Convolutional Neural Network
— R-CNN: Region CNN
— ConvNet: Convolutional Neural Network
— FCN: Fully Connected Network
— DNN: Deep Neural Network

— SVM: Support Vector Machine
— SGD: Stochastic Gradient Descent
— MCMC: Monte Carlo Markov Chain
— GAN: Generative Adversarial Networks
— VAE: Variational AutoEncoders
— IoU: Intersection over Union
— ROC: Receiver Operating Characteristic
— AUC: Area Under the Curve
— MCMC: Monte Carlo Markov Chain
— UMAP Uniform Manifold Approximation and Projection

Résumé en français

Contexte de la thèse

La dosimétrie biologique

La dosimétrie est la discipline qui vise à estimer les doses de radiation ionisantes reçues par les être vivants ou les objets. Dans le cas où un dosimètre est disponible, l'estimation peut en général être réalisée de manière rapide et fiable par lecture directe. Si le contexte d'exposition est connu, la dose peut être reconstruite par simulation Monte-Carlo. Cependant, le scénario d'exposition peut être inconnu dans le cas d'une exposition accidentelle. Dans ce contexte, la dosimétrie biologique propose des méthodes alternatives pour estimer ces doses. Les rayonnements ionisants ayant des effets délétères sur les cellules du corps humain, les traces résiduelles peuvent être exploitées pour estimer la dose initialement reçue.

Actuellement, la méthodologie recommandée par l'Agence Internationale pour l'Energie Atomique (AIEA) en matière de dosimétrie biologique cytogénétique est le comptage de chromosomes dicentriques dans les lymphocytes périphériques. Plus exactement, le biomarqueur d'intérêt est le nombre moyen de chromosomes dicentriques par cellule. Même si les rayonnements ionisants produisent d'autres types d'aberration chromosomique, comme les fragments de chromosome ou les chromosomes en anneau, le comptage de ces derniers ne fait pas partie du protocole défini par l'AIEA. Au cours d'un examen, un échantillon sanguin est collecté, et les cellules sanguines sont mises en culture pendant 48 heures. La mitose est inhibée durant la métaphase avec du Demelcocine [1]. Ensuite, l'échantillon sanguin est fixé sur une lame, coloré en Giemsa et examiné avec un microscope optique. Le Giemsa est un colorant couramment utilisé en histologie, composé de bleu de méthylène et d'éosine. Il adhère spécifiquement au groupes phosphate de l'ADN. Par conséquent, c'est un outil couramment utilisé en cytogénétique pour la visualisa-

1. Le Demecolcine (également connu sous le nom de colcemid) est un médicament utilisé en chimiothérapie. Le Demelcocine inhibe la mitose pendant la métaphase en limitant la formation des microtubules. Ce médicament permet de synchroniser les cellules cancéreuses en métaphase afin de maximiser leur radio-sensibilité. *Source*: https://en.wikipedia.org/wiki/Demecolcine.

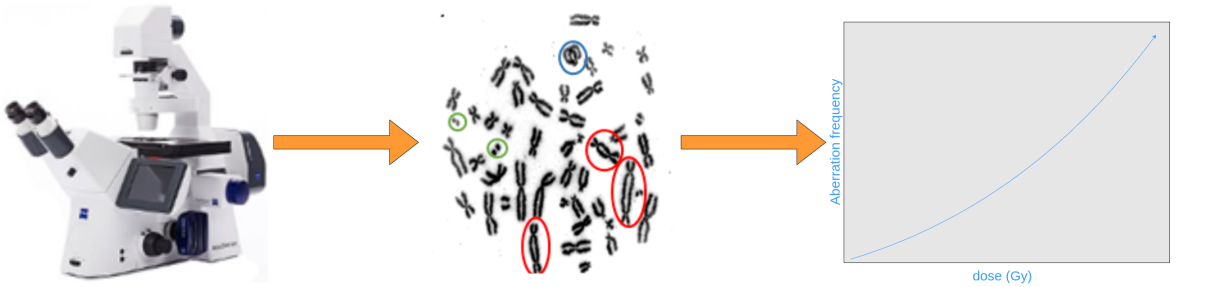


Figure 1 – Chronologie d'une estimation de dose de rayonnement ionisant en dosimétrie biologique. Un échantillon sanguin est mis en culture pendant 48h, et la mitose est inhibée durant la métaphase avec du Demecolcine. Les microscopes modernes permettent une acquisition automatisée des métaphases, comme montré en a). Les experts biologistes comptent les chromosomes dicentriques (entouré en rouge) et identifient également les fragments (en vert) et les anneaux-centriques (en bleu), voir b). Finalement, le nombre moyen d'aberrations par lymphocyte permet d'estimer la dose, comme montré en c).

tion des chromosomes. Durant l'étape de métaphase, les chromosomes des lymphocytes sont visibles. Pour cette raison, une image des 23 paires de chromosomes appartenant à un lymphocyte en métaphase sera appelée une "métaphase" par les spécialistes du domaine, un terme que nous reprenons dans la suite du manuscrit. Un expert biologiste examine ensuite la métaphase afin de compter les chromosomes dicentriques. Ces aberrations sont rares, même pour une exposition à une dose élevée. Par exemple, en l'absence d'exposition, environ une cellule sur mille contient un chromosome dicentrique. Une fois ce nombre moyen d'aberrations calculé, une courbe de calibration permet de déterminer la dose correspondante. La Figure 1 présente un résumé du protocole utilisé en dosimétrie biologique par cytogénétique.

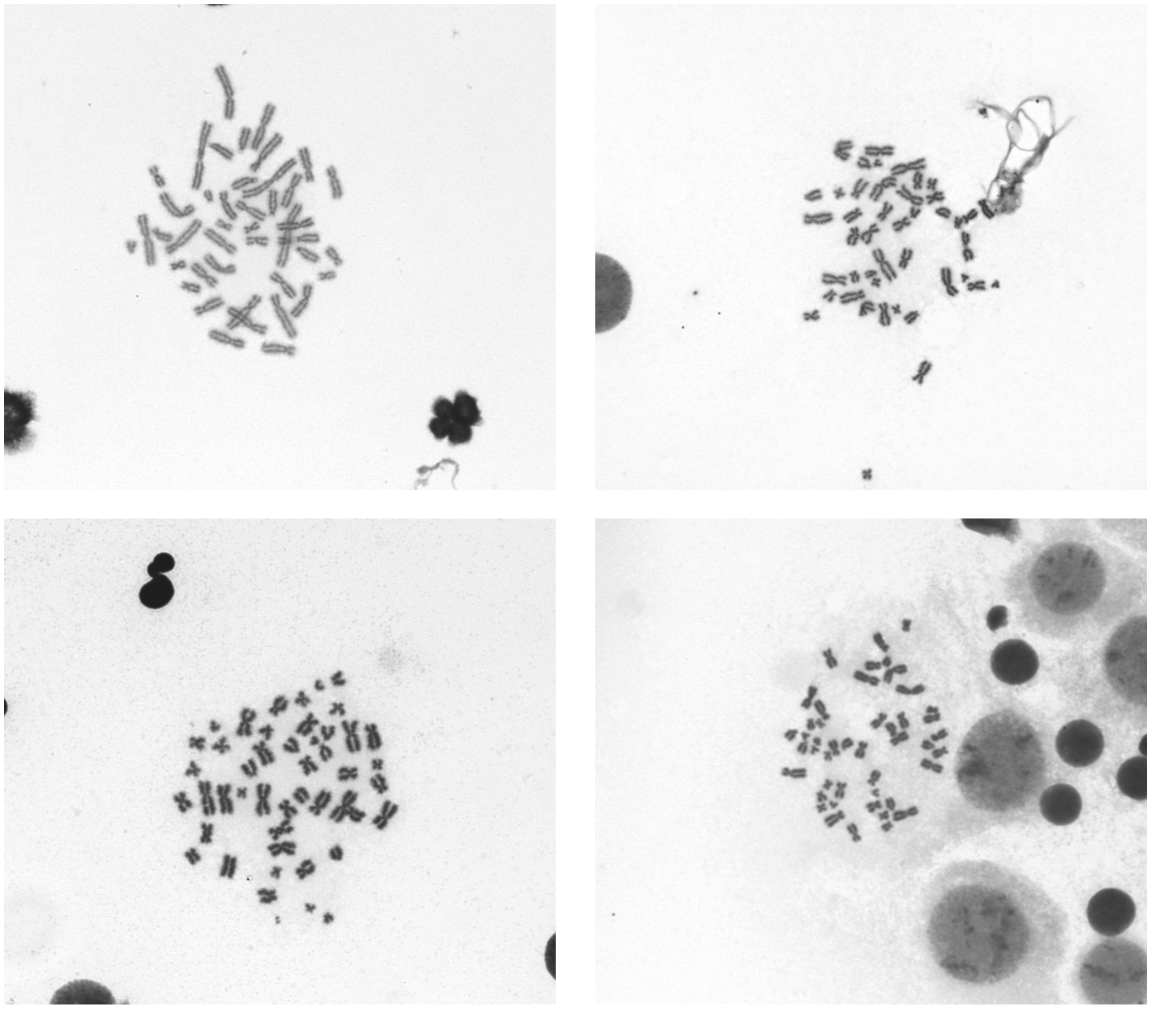
Une formation de plusieurs mois est nécessaire pour habiliter une personne à compter fiablement les aberrations chromosomiques. Les chromosomes sont des objets déformables, et certains chromosomes monocentriques peuvent être déformés de telle sorte qu'ils ressemblent à des chromosomes dicentriques. Si une population d'individus importante est exposée à une source de radiation, le comptage manuel des aberrations pour chaque individu est impossible. Le développement d'une solution automatisée est l'objet principal de cette thèse pluridisciplinaire, réalisée dans le cadre d'une collaboration entre deux laboratoires: le LRAcc (IRSN) et SERPICO (Inria). Le Laboratoire pour la Radiobiologie des expositions Accidentelles (LRAcc) est l'un des rares laboratoires français détenant une expertise en dosimétrie biologique. Il fait partie de l'Institut pour la Radioprotection et la

Sûreté Nucléaire (IRSN), et est considéré comme l'expert de référence pour la dosimétrie biologique en conditions accidentelles. L'expertise en matière d'imagerie biologique est fournie par l'équipe Space-timE RePresentation, Imaging and cellular dynamics of molecular COmplexes (SERPICO) de l'Institut National pour la Recherche en Informatique et Automatique (Inria) de Rennes.

Cahier des charges pour un outil de dosimétrie biologique automatisée

La conception d'un système de détection automatique de chromosomes dicentriques présente de nombreuses difficultés. Par exemple, l'adhésion du colorant Giemsa au chromosomes présente une certaine variabilité, ce qui affectera le contraste entre les chromosomes et l'arrière plan. Même si la mitose est inhibée en métaphase par le Demelcocine, les cellules ne sont pas toutes figées au même instant, ce qui induit une certaine variabilité inter-cellulaire. Cela se traduit par des variations de longueur, d'épaisseur et de texture des chromosomes (voir Figure 2 B)). Lorsque l'échantillon sanguin est fixé à la lame, certaines cellules sont proches du rétroéclairage du microscope et d'autres sont plus éloignées, ce qui induit des variations d'illumination dans les images. L'étalement de l'échantillon sur la lame est par ailleurs un procédé aléatoire, sans garantie de produire des métaphases suffisamment étalées pour pouvoir compter les chromosomes et identifier les dicentriques (voir Figure 2 A)). Enfin, la déformabilité des chromosomes produit une variabilité morphologique extrême qui complique la classification automatique des chromosomes. Un système de détection automatique des aberrations se doit d'être robuste au facteurs de variation visible sur la Figure 2.

Finalement, le taux de base d'aberrations impose un taux de faux positifs très faible. Tout taux de faux positifs dépassant 1 pour 1000 conduira nécessairement à une surestimation de la dose. Par conséquent, le système doit être capable de mesurer l'incertitude de prédiction: les faux positifs doivent être interprétés comme des détections incertaines. Sur la base de ces constats, cette thèse vise à utiliser les avancées récentes de l'apprentissage statistique et les réseaux de convolution profond pour élaborer un système de détection automatique d'aberrations plus performant que les systèmes commerciaux actuels.



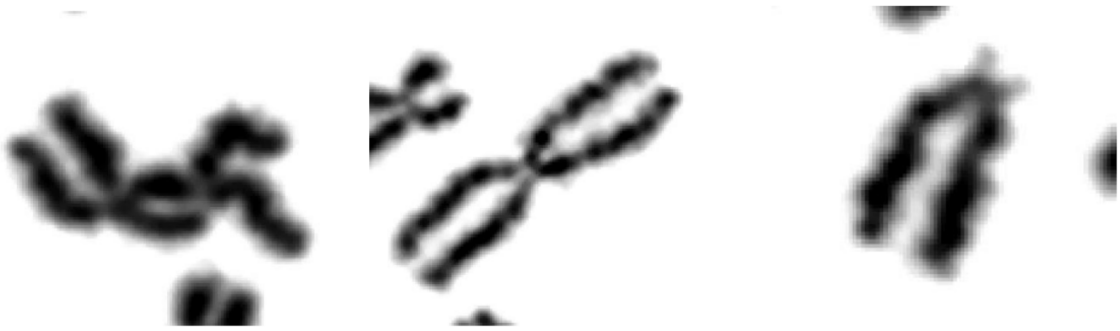


Figure 2 – A) Figure montrant les variations possibles en terme d'illumination, de longueur de chromosomes et de présence de noyaux et débris cellulaires B) Figure montrant les variations possibles en terme de longueur de chromosome, de position du centromere et de repliement des chromatides

L'apprentissage profond en vision par ordinateur

Dans un article fondateur pour la vision par ordinateur, Krizhevsky *et al.* [1] on démontré il y a une dizaine d'années qu'il était possible d'obtenir des performances élevées sur ImageNet [2] (un jeu de données de classification) avec un réseau de neurones convolutionnel (appelé AlexNet) dont les paramètres sont appris en minimisant une entropie croisée via un algorithme de descente de gradient stochastique. Depuis, les réseaux convolutifs ont fait l'objet de recherches extrêmement actives dans la communauté scientifique de la vision par ordinateur pour résoudre des tâches comme la classification, la détection d'objets, la segmentation d'image, l'estimation du flot optique, etc.

Les grandes bases de données annotées telles que ImageNet [2], Common Objects in Context (COCO) [3] ou encore Pascal VOC [4] se sont révélées cruciales pour la recherche sur les réseaux convolutionnels. En effet, ces modèles nécessitent en général d'importantes quantités de données pour atteindre un haut niveau de performance. La performance progresse presque continûment avec la taille de la base de données, ce qui n'était pas nécessairement une caractéristique des modèles disponibles auparavant. Finalement, ces bases de données permettent d'évaluer différentes architectures de manière objective, et facilite les comparaisons entre les méthodes proposées par la communauté scientifique.

Cependant, ces grandes bases de données annotées sont plus rares dans le contexte de l'imagerie biomédicale pour plusieurs raisons. Les annotations sont particulièrement chronophages (et donc coûteuses) car les objets à détecter dans les images de microscopie ou de scanner sont souvent ambigus. Construire des grandes de bases de données annotées peut également poser des problèmes d'ordre juridique relatifs à la préservation de la vie privée. La diversité des modalités d'imagerie et des contextes biologiques rend également la construction d'une base de données "universelle" (similaire à ImageNet) impossible. Par conséquent, pré-entraîner les réseaux convolutifs sur des données sans annotation est une préoccupation majeure en apprentissage machine. Ce besoin est au coeur d'un champ de recherche appelé "apprentissage auto-supervisé", ou *Self Supervised Learning* (SSL). En apprentissage auto-supervisé, l'objectif est de construire des critères d'apprentissage qui ne dépendent d'aucune annotation. Une fois qu'un modèle est pré-entraîné sur une grande base de données sans annotation (en général plus facile à obtenir), il est possible d'utiliser ce modèle comme initialisation pour un apprentissage sur une base de données supervisée. Ce principe est appelée apprentissage par transfert (*transfer learning*).

Un autre problème majeur des les réseaux convolutifs est la quantification des incertitudes. En effet, pour une tâche de classification, la probabilité prédite ne correspond pas

au taux d'erreur observé sur cette classe [5]. Par exemple, un modèle prédisant une classe avec une confiance moyenne de 99% aura en général un taux d'erreur supérieur a 1% sur cette classe. En d'autre mots, la probabilité prédite ne peut pas être utilisée comme indicateur de confiance du modèle. Cette caractéristique des modèles profonds est particulièrement problématique dans le contexte de l'imagerie médicale, ou les conséquences d'une erreur du modèle peuvent être très importantes. Construire des indicateurs de confiance fiables est donc un domaine de recherche actif. Le paradigme Bayésien est une manière naturelle de construire ces indicateurs de confiance, mais l'échantillonnage de la distribution *a posteriori* des poids avec les méthodes habituellement utilisées (*e.g.* méthode de Monte Carlo) n'est pas une option réaliste pour les réseaux convolutionnels profonds, en raison du grand nombre de paramètres. Une approche alternative est celle proposée par l'apprentissage profond Bayésien approximatif. La distribution *a posteriori* est supposée Gaussienne, et plusieurs techniques ont été développées pour estimer les paramètres sous-jacents. Par exemple, dans [6] les itérés de la descente de gradient sont considérés comme des échantillons de la distribution *a posteriori* (un résultat démontré dans [7]), et utilisés pour estimer la moyenne et la matrice de covariance de la distribution *a posteriori* approximative. Cette distribution peut être échantillonnée pour construire un ensemble de modèles afin de quantifier l'incertitude de la prédiction.

Contributions de cette thèse

La première partie de cette thèse (Chapitres 1 et 2) est consacrée à une présentation (probablement incomplète) de l'état de l'art de l'apprentissage profond en vision par ordinateur et en dosimétrie biologique. Dans le Chapitre 1, nous présentons les éléments de base de l'apprentissage profond, des réseaux convolutionnels, de l'apprentissage auto-supervisé et de l'apprentissage profond Bayésien approximatif. Dans le Chapitre 2 nous présentons un état de l'art de la dosimétrie biologique automatisée. Dans ce contexte, le logiciel Automated Dicentric Chromosome Identifier (ADCI) fait l'objet d'une attention particulière, car c'est la solution commerciale dont le fonctionnement est le mieux documenté, à travers une série d'articles publiés de 2012 à 2019. Nous présentons également l'évaluation de DCScore (une autre solution commerciale de détection de chromosomes dicentriques) dans des contextes variés. Finalement, des avancées très récentes en matière de dosimétrie automatisée, reposant sur l'apprentissage profond sont présentées. En décrivant le fonctionnement de ces solutions, ce chapitre permet également de présenter les points

de difficulté majeurs dans la conception d'un tel système, dont les prémices sont présentés dans le Chapitre 3.

Les premières tentatives pour réaliser un tel système de détection d'aberrations chromosomiques reposent sur un schéma en deux étapes: la détection de tous les chromosomes, puis la classification de ces derniers en deux classes (chromosomes monocentriques et chromosomes dicentriques). Dans ce système analogue à ADCI, nous utilisons un réseau convolutionnel profond appelé ResNet [8] comme classifieur monocentrique vs dicentrique. Ce classifieur est entraîné sur une base de données de petites images annotées (appelées patch). Ce classifieur permet d'atteindre des niveaux de performance élevés sur cette tâche de classification binaire sans grande difficulté. Cependant, nous ne sommes pas parvenus à reproduire ce niveau de performance avec des patches extraits par des modèles de détection d'objets simples, comme un clustering K-Means. Nous avons identifié deux raisons principales à cet échec: i) la base de données n'était pas suffisamment représentative, ii) le modèle de détection d'objets n'était pas suffisamment performant. Les patches à classer contenaient parfois plusieurs chromosomes, ou des fragments de chromosomes entiers qu'il n'était pas possible de classer correctement. La détection imparfaite des chromosomes dégrade *in fine* assez vite la performance de classification

Nous avons également tenté d'utiliser cette base de données de patches pour apprendre à simuler des chromosomes afin de pré-entraîner des modèles de détection d'objets plus sophistiqués (basés sur l'apprentissage profond) sur des données synthétiques. Construire un simulateur de chromosomes performant s'est révélé particulièrement difficile en définitive, et nous n'avons pas réussi à concevoir une solution véritablement satisfaisante. Afin de tout de même pouvoir évaluer le comportement de modèles de détection d'objets sur des données simulées, nous avons construit des métaphases synthétiques annotées (pour lesquelles les chromosomes dicentriques sont localisés par des boites englobantes) à partir de "squelettes" de chromosomes. Même si ces métaphases synthétiques ne sont pas particulièrement réalistes, nous les avons utilisées pour évaluer Faster R-CNN [9] dans un contexte de détection d'objets rares (tel que la dosimétrie biologique). La rareté des chromosomes dicentriques induit une dépendance forte entre la performance du modèle et le nombre d'images utilisées pour entraîner le modèle. En effet, pour les doses les plus faibles, une base de données d'apprentissage comprenant plusieurs milliers d'images ne contient que quelques exemples de chromosomes dicentriques différents. Nous avons également constaté qu'un modèle entraîné sur des données associées à un scénario de dose forte (i.e., avec un nombre moyen élevé de chromosomes dicentriques par métaphase) s'avérait moins

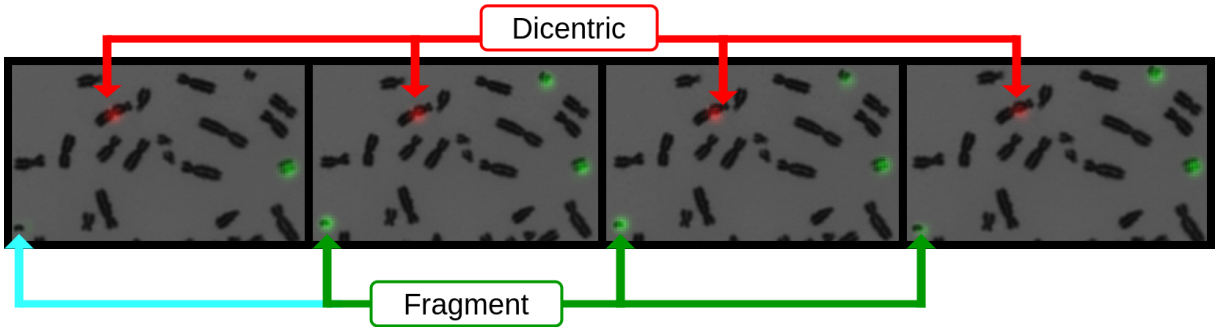


Figure 3 – Prédictions d'un ensemble de quatres modèles différents. Le premier modèle ne détecte pas le fragment de chromosome (en vert) en bas à gauche de l'image. Les trois autres modèles de l'ensemble le détectent correctement. Les quatres modèles prédisent le chromosome dicentrique près du centre de l'image correctement.

performant lorsqu'il est évalué sur des données correspondant à un scénario de dose plus faible. Sur-représenter les chromosomes dicentriques pour faciliter l'apprentissage tend à augmenter le nombre de faux positifs pour les faibles doses. La simulation nous a permis d'identifier ces problèmes et d'élaborer une autre stratégie plus efficace.

La contribution principale de cette thèse est une preuve de concept en dosimétrie automatisée qui exploite une grande base de données annotée, pour laquelle les centres de chromosomes dicentriques et de fragments sont localisés dans les métaphases. Cette preuve de concept repose sur l'architecture Unet [10], un terme de régularisation encourageant une prédiction parcimonieuse [11] et l'algorithme d'optimisation Adam [12]. Notre approche utilise également les résultats récents d'Izmailov *et al.* [13] et de Mandt *et al.* [7] en matière d'agrégation d'itérés de la descente de gradient collectés pendant l'apprentissage. Les paramètres du modèle sont collectés à intervalles réguliers au cours de l'apprentissage, et nous construisons un ensemble en tirant au hasard plusieurs vecteurs de paramètres. Chacun de ces modèles atteint un haut niveau de performance, mais l'ensemble est également suffisamment diversifié (comme le montre la Figure 3) pour qu'il soit possible d'être plus performant en agrégant les prédictions de plusieurs modèles qu'en sélectionnant un seul modèle. Les aberrations sont acceptées ou rejetées sur la base de l'accord entre les différents modèles de l'ensemble. Finalement, en utilisant certaines connaissances *a priori* sur la dosimétrie biologique, nous parvenons à dépasser les problèmes de transfert entre les scénarios de dose identifiés sur les données simulées, et nous démontrons que notre modèle parvient à estimer des courbes de calibration très satisfaisantes sur un jeu de données de test non-annoté de plus de 21 000 images.

Organisation du manuscript

Le Chapitre 1 présente quelques articles de référence de l'apprentissage profond pour la vision par ordinateur. Ce Chapitre introduit les bases des réseaux de neurones ainsi que des réseaux de convolutions, de la rétropropagation et des méthodes du premier ordre utilisées pour entraîner cette classe de modèles. Les principales architectures en apprentissage auto-supervisé sont également présentées, telles que les autoencodeurs et les approches basées sur la consistence. Finalement, ce chapitre décrit les grands principes de l'apprentissage Bayésien profond approximatif, car c'est une composante majeure de la preuve de concept présentée dans le Chapitre 4.

Le Chapitre 2 est un état de l'art en dosimétrie biologique automatisée utilisant des images colorées au Giemsa. Nous passons en revue deux logiciels conçus pour compter automatiquement les chromosomes dicentriques. Tout d'abord, ADCI est une solution commerciale développée par Cytognomix, une entreprise canadienne. Cette solution est un pipeline "détection-classification" qui n'utilise pas de réseaux convolutionnels. Les articles présentant les composants de cette solution donnent une idée précise de son mode de fonctionnement, ce qui est rare dans le contexte des solutions commerciales en imagerie biomédicale. Ensuite, nous présentons une série d'articles issus principalement du LRAcc qui évaluent DCScore, une autre solution commerciale de détection de chromosomes dicentriques proposée par l'entreprise allemande MetaSystems. Même si DCScore n'est pas suffisamment performant pour être utilisé de manière complètement automatisée, ce logiciel peut être utilisé de manière semi-automatique, les chromosomes dicentriques détectés font l'objet d'une revue manuelle. Finalement, nous présentons les solutions automatisées existantes qui reposent sur le principe de l'apprentissage profond.

Le Chapitre 3 décrit nos travaux portant sur la reconnaissance de chromosomes individuels. Trois approches principales sont décrites: la classification de chromosomes, la synthèse de chromosomes et l'évaluation de modèle de détection d'objets sur métaphases synthétiques. Entraîner un classifieur de chromosomes est relativement aisé avec l'architecture ResNet [8]. Cependant, la performance de ce classifieur sur des vraies métaphases dépend de la représentativité de la base d'apprentissage, ainsi que de la qualité de notre détecteur de chromosomes, et construire un pipeline efficace est une tâche difficile. Finalement, même si nous ne sommes pas parvenus a construire un simulateur de chromosome performant, l'entraînement de Faster R-CNN [9] sur une base de données de "squelette" de chromosomes nous a permis d'étudier le comportement de ce modèle en fonction du volume de

données.

Dans le Chapitre 4 nous présentons notre détecteur de chromosomes dicentrique, qui repose sur une architecture Unet, une méthode d'optimisation stochastique du premier ordre (Adam), un terme de régularisation encourageant des prédictions parcimonieuses et une technique d'agrégation de modèles. L'agrégation permet de quantifier en particulier le niveau de confiance associé à une détection, et de rejeter les détections peu confiantes. Notre méthode d'agrégation repose sur des paramètres interprétables, et atteint un haut niveau de performance en considérant à la fois les métriques de détection d'objets et l'estimation de courbes de calibration.

Communication scientifique

Notre preuve de concept pour un détecteur automatique de chromosomes dicentriques a été décrite dans un article intitulé "Ensembling Unets, sparse representation and low dimensional visualization for rare chromosomal aberration detection in light microscopy images" (https://doi.org/10.1101/2023.09.11.557124), en cours de soumission à une revue internationale. Cette preuve de concept a également été présentée lors de l'*International Conference on Radiation Research* à Montréal en août 2023.

De juin à septembre 2022, j'ai réalisé un séjour scientifique dans le département *Engineering Physics* de l'université McMaster au Canada, sous la direction de Qiyin Fang. Ce séjour avait pour objectif d'implémenter des méthodes d'estimation du flot optique pour l'imagerie cellulaire microfluidique, dans le cadre d'une thèse (Tianqi Hong). Cette méthode d'estimation du flot optique réutilise le régulariseur parcimonieux utilisé pour la preuve de concept du Chapitre 4. Ce séjour a débouché sur la rédaction d'un pre-print qui fait l'objet de l'Annexe A de cette thèse.

INTRODUCTION

Context and motivations

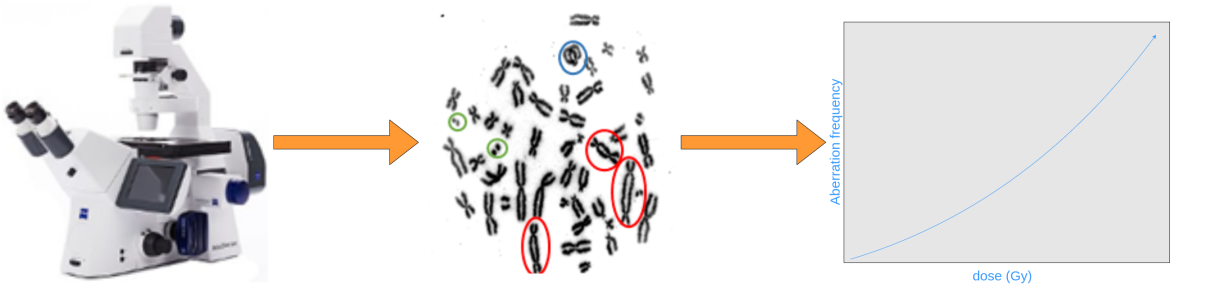
Biological dosimetry

Dosimetry is the branch of health physics dealing with estimating ionizing radiation doses received by living organisms. If a dosimeter is available, the dose can be read directly with this tool. Even if a dosimeter is not available, the dose can be reconstructed using Monte-Carlo simulations if the exposition context is known. However, if the exposition is accidental the exposition context is usually unknown. In this case, biological dosimetry provides alternative tools to estimate doses of ionizing radiation. As radiation has damaging effects on human cells, several biomarkers can be used to estimate this dose only using information available after the exposition.

The current gold standard in cytogenetic biological dosimetry, defined by the International Atomic Energy Agency (IAEA) is Dicentric Chromosome (DC) scoring [14] in peripheral blood lymphocytes. More precisely, the biomarker of interest is the average number of DC per cell, also called dicentric yield. While radiation induces additional aberrations like chromatid fragments (usually abbreviated as "fragment") or ring-centric chromosomes (usually abbreviated as "rings"), they are currently not considered in the protocol defined by the IAEA.

To estimate a dose, a blood sample is taken from a patient and blood cells are grown. After 48 hours, cell mitosis is inhibited with Demecolcine [2], the blood sample is stained with Giemsa, mounted on a slide, and examinated with a brightfield microscope. Giemsa is a well known histological stain composed of methylene blue and eosin. It binds specifically to the phosphate groups of DNA. Because of this, it is commonly used in cytogenetics to visualize chromosomes. With Giemsa staining, the chromosomes of lymphocytes are visible during metaphase. Because of this, images of chromosomes belonging to a metaphase cell

2. "*Demecolcine (also known as colcemid) is a drug used in chemotherapy* [...] *During cell division, demecolcine inhibits mitosis at metaphase by inhibiting spindle formation. Medically, demecolcine has been used to improve the results of cancer radiotherapy by synchronising tumour cells at metaphase, the radiosensitive stage of the cell cycle*". *Source*: https://en.wikipedia.org/wiki/Demecolcine.



a) metaphase acquisition b) aberration counting c) dose estimation with calibration curve

Figure 4 – Chronology of a biological dosimetry-based estimation of an ionizing radiation dose. First, a blood sample is taken and cells are grown for 48 hours, Demecolcine is used to stop cells in the metaphase stage of mitosis. Metaphase acquisition is automated with modern microscopy systems, as shown in a), and specalists count DCs (circled in red), fragments (circled in green) and ring-centric chromosomes (circled in blue) as displayed in b). The proportion of chromosomal aberration is linked to the ionizing radiation dose through a linear-quadratic relationship, see c).

are called "metaphases" by biological dosimetry experts. A trained specialist examines the metaphase and detects DCs.

While any cell contains 23 chromosome pairs, *i.e.* 46 chromosomes, most chromosomes are healthy Monocentric Chromosomes (MCs), even at high doses, and DCs are rare events. The base rate of aberration (when there is no exposition) is around 1 DC per 1000 lymphocytes. Once the dicentric yield is computed, a calibration curve provides the underlying dose. Figure 2.2 provides an overview of the chronologized steps of biological dosimetry.

Training is needed to become proficient at aberration detection, as separating DCs from normal MCs is difficult. Chromosomes are deformable objects and some MCs may twist and become indistinguishable from a DC for an untrained expert. In the context of a large scale exposition, manual aberration scoring is widely acknowledged to be a bottleneck and automated solutions are required.

This thesis aims to explore recent advances in deep learning for computer vision to improve on currently available commercial Automated Dicentric Scoring (ADS) systems. This PhD has been a multidisciplinary endeavour and was completed between two labs. The Laboratory for Radiobiology of Accidental Expositions (LRAcc) is one of the main french laboratories researching biological dosimetry. It is part of the Institute for Radioprotection and Nuclear Safety (IRSN). LRAcc is focused on dose estimation in irradiation settings. The interest of LRAcc in automating DC detection led to the funding and su-

pervision of this thesis.

As the LRAcc does not have any specific expertise in biomedical imaging, it was provided by the Space-timE RePresentation, Imaging and cellular dynamics of molecular COmplexes (SERPICO) team at the National Institute for Research in Informatics and Automation (Inria).

Challenges and requirements of an automated DC detector

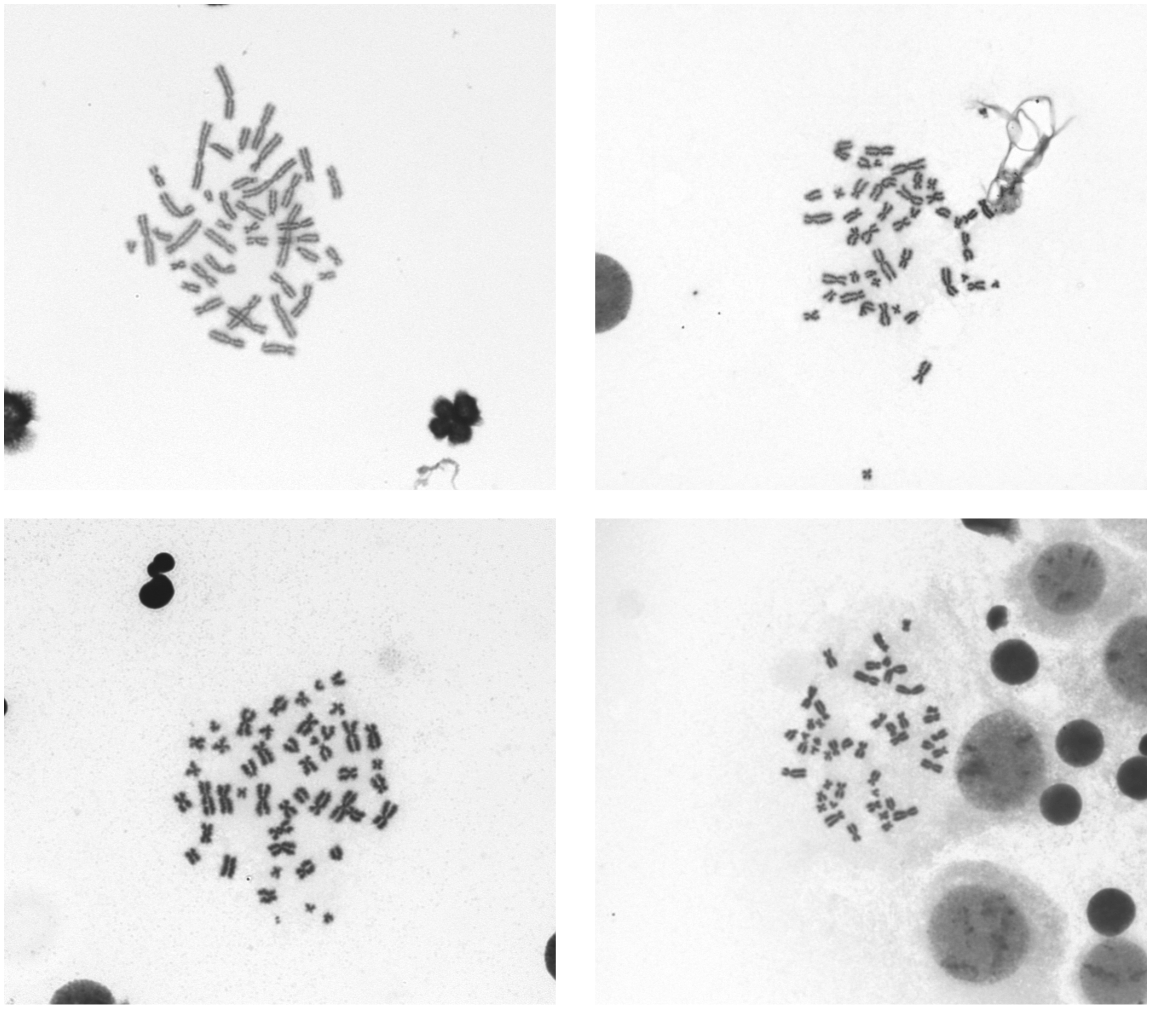
It is well established that building an ADS system is challenging. The Giemsa staining process is inhomogeneous and staining does not bind to every chromosome evenly. This leads to variations in intensities values of image pixels. While Demelcocine is used to stop mitosis, cells are stopped at various degrees of metaphase which leads to variation in chromosome length, width and texture. As the blood sample is spread on the glass slide, some cells are very close to the center of the slide, while some others are closer to the edges. This leads to variations in illumination, and therefore to variations in pixel intensity values.

Furthermore, chromosome spread is inherently random and there is no way to guarantee that metaphases are sufficiently spread for chromosome counting and DC detection, as is shown in Figure 5 A). Finally, the deformability of chromosomes generates extreme morphological variety (as shown in Figure 5 B)), which complicates automated chromosome classification. Any effective ADS system needs to be robust to the variations in the chemical process, the changes in illumination and the morphological variations of chromosomes, as shown in Figure 5.

Finally, the base rate of chromosomal aberration imposes extremely strict constraints on the rate of false positives. Any system that produces more than one erroneous detection every thousand cell overestimates the dose of a patient that was not irradiated. Because of this, model uncertainty should be quantified: false positives should be low-confidence detections. Accordingly, we investigated recent advances in deep learning to improve on the performance of currently available commercial ADS solutions.

Deep learning for computer vision

Since Krizhevsky *et al.* [1] demonstrated state-of-the-art performance on the ImageNet image classification dataset [2] using a Convolutional Network (ConvNet) model called AlexNet trained with Stochastic Gradient Descent (SGD), ConvNets have been



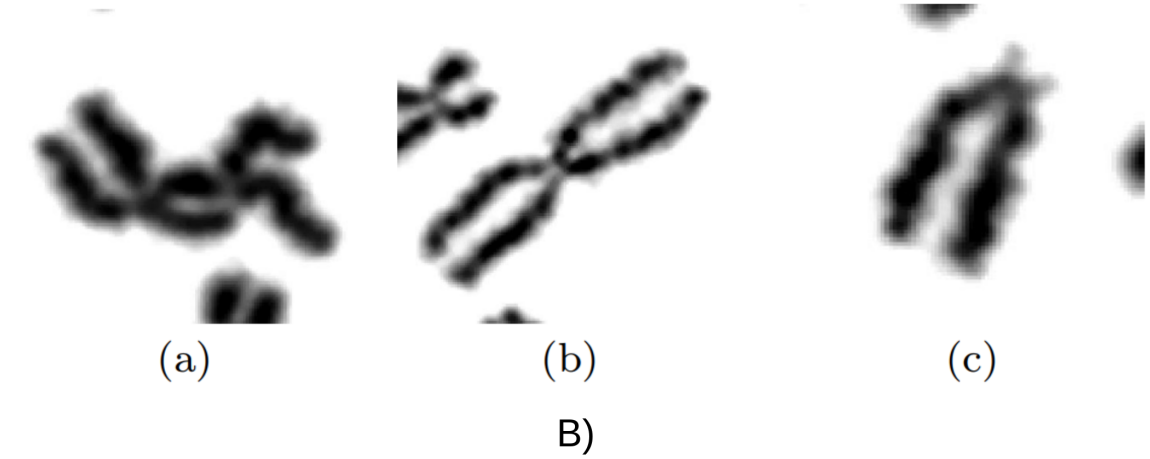


Figure 5 – A) Figure showcasing possible variations in illumination, chromosome length, chromosome debris and presence of nuclei and debris B) Figure showcasing variations in chromosome length, centromere location and folding of chromatides

an extremely active area of research in computer vision. This class of model has been used for tasks like object detection, object segmentation, image classification, optical flow estimation, etc.

Large annotated benchmark datasets like ImageNet [2], Common Objects in Context (COCO) [3] and Pascal VOC [4] have made research on ConvNets for computer vision considerably easier. First, a large dataset is usually needed to reach a high level of performance with ConvNets. It is worth noting that performance keeps improving as the dataset grows, which was not always a given with previous image analysis models. Second, those datasets are used as standardized benchmarks, which makes objective comparison between methods possible.

However, biomedical imaging lacks benchmark datasets like those available in computer vision for several reasons. Labels are much more expensive as they usually require the expertise of trained medical experts and the relevant objects are harder to locate and classify in microscopy or scanner images. Furthermore, curating such a dataset may run into privacy issues. Finally, the wide variety of imaging modalities and biological settings makes building an "universal" dataset unfeasible.

Therefore, the ability to pretrain ConvNets on un-labelled data is crucial. This requirement has motivated works in Self-Supervised Learning (SSL) [15]. SSL focuses on designing training criterion that do not use any label. Once a model is trained on a large unlabelled dataset it can be fine-tuned on a smaller labelled dataset, a process called "transfer learning".

Another issue facing ConvNets is uncertainty quantification. ConvNets trained on classification tasks are usually overconfident [5] which means that the predicted probability does not match the error rate. A ConvNet that predicts a class with a probability of 99% has an error rate greater than 1% on this class. In other words, predicted probability cannot be used as a metric for model confidence. This unwanted characteristic of deep models is especially problematic in medical imaging, where the consequences of a classification error can be catastrophic.

Accordingly, building accurate confidence metrics for ConvNets becomes an active area of research. The Bayesian framework is a natural way of modelling model uncertainty, but sampling the posterior distribution over the networks weight with conventional methods (like Monte Carlo Markov Chain) is usually computationally intractable, because of the very large number of parameters in modern ConvNets.

Alternative solutions have been investigated. In approximate Bayesian deep learning,

the posterior distribution over the weights is usually assumed to be Gaussian, and several techniques have been developped to estimate the parameters of this distribution. A noteworthy example is the one provided by Maddox *et al.* in [6]. In this paper successive iterates of the network weights during SGD are taken as approximate samples of the posterior distribution over the weights, as suggested by Mandt *et al.* in [7]. Those samples are used to estimate the mean and the variance of the approximate posterior distribution. This approximate posterior distribution can be used to sample an ensemble of models which can be used for uncertainty quantification.

Contribution of this thesis

The first part of this thesis (Chapter 1 and 2) provides a introduction to deep learning and automated biological dosimetry. Chapter 1 includes the basics of deep learning, ConvNets, unsupervised learning, and approximate Bayesian deep learning. In Chapter 2, we give an overview of automated biological dosimetry. Most importantly, we describe the Automated Dicentric Chromosome Identifier (ADCI), a current comercially available automated dosimetry solution. The components of this solution are described in several papers published between 2012 and 2019.

We present a few papers written by researchers at LRAcc that evaluate DCScore in various contexts. Finally, we mention two papers implementing ADS systems based on deep learning. By describing current ADS solutions, Chapter 2 provides an overview of the major technical difficulties encountered in the design of automated biodosimetry software.

In our first attempts at designing an ADS system, we tried to reproduce a two-step, detection-classification ADS pipeline with a ResNet-based [8] MC-DC classifier. We used a dataset of labelled chromosome patches to train the ResNet binary classifier used in the second step.

While reaching a very high level of performance on this classification step was relatively easy to achieve, reproducing this performance on real world data (*i.e.*, chromosomes extracted from arbitrary metaphases) proved to be much harder. We identified two main reasons for this failure: our dataset was not representative enough, and our object detection model was not good enough, leading to ambiguous patches that could not be classified accurately. In other words, the performance of the detection and classification steps were strongly dependent.

Furthermore, we attempted to synthesize images of chromosomes, so that we could

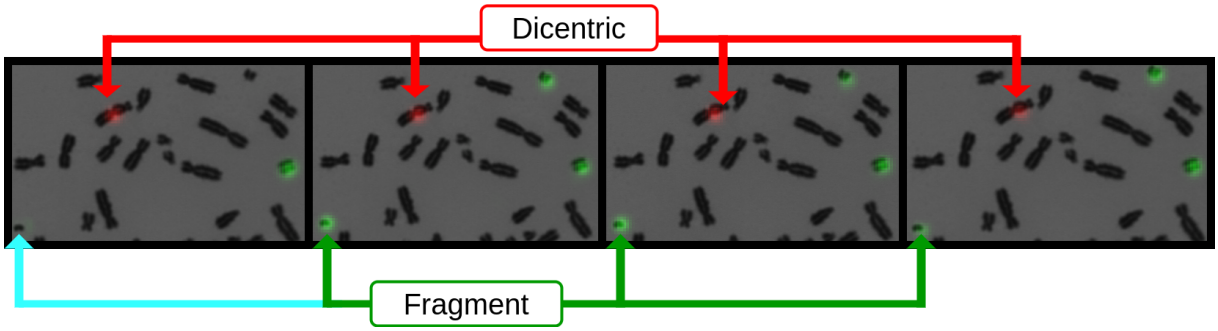


Figure 6 – Prediction diversity for a set of four different models. The first model does not predict the fragment in the bottom left corner, but the three next model do. All four models predict the DC near the center of the image correctly.

pre-train object detection models on simulated data. Designing a performant chromosome simulator proved to be a challenging task and we failed to find a satisfying solution.

In order to investigate the performance of Faster R-CNN, a relatively modern, deep learning-based object detection model, we simulated labelled metaphase images (metaphases where DCs are located with bounding boxes) using chromosome "skeletons". Because of the under-representation of DCs model performance exhibited a strong dependence to the number of training images.

In a low dose scenario, a dataset containing several thousands images might only contain a few examples of DCs. Furthermore, models trained on data corresponding to a high dose scenario (with a high average number of aberrations per cell) performed worse on "low-dose" data. This means that oversampling images containing large number of aberrations to circumvent their under-representation leads to an increase in false positives.

While we did not build a viable ADS system with our chromosome dataset and did not learn to synthesize realistic data, the insights of this simulation study have been helpful to design an alternative strategy in the next chapter.

The main contribution of this thesis is a proof of concept for DC detection and dose estimation. It is trained on large labelled dataset which was built over the course of several months at LRAcc. In this dataset, the centers of all chromosomal aberrations (including fragments and rings) are located in metaphases. Our solution is based on the Unet architecture [10], a sparsity-promoting regularizing term [11] and gradient-based training with Adam [12].

Furthermore, we benefit from recent results in DNN agregation by Izmailov *et al.* [13], based on theoretical insight from Mandt *et al.* [7]. We build an ensemble from a random

sample of the checkpoints collected at the end of each epoch. While each model of this ensemble reaches a high level of performance, it is also highly diverse as shown in Figure 6.

This can be used to improve performance beyond the single-model baseline by accepting or rejecting aberration detections based on the agreement between the models of the ensemble. Finally, using *a priori* knowledge on biological dosimetry, we overcome the issues related to the transfer between different dose scenarios already identified on simulated data. We show that our model is able to estimate very satisfying calibration curves on a separate unlabelled testing dataset comprised of over 21 000 images.

Organization of the manuscript

In Chapter 1, we provide a review of relevant references in the deep learning literature for computer vision. The basics of neural networks, ConvNets, backpropagation, and gradient-based training are introduced.

A few important references of the similarity-based unsupervised learning literature are provided as examples of the dominant approach in the current state of the art. Finally, the approximate Bayesian deep learning literature is given an introduction, as it is a major component of the proof of concept introduced in Chapter 3.

In Chapter 2, the field of automated biological dosimetry in Giemsa-stained images is introduced. As biological dosimetry remains a niche application in computer vision there are only a few competing ADS solutions and to the best of our knowledge, all of them being commercial software. First, a thorough overview of the Automated Dicentric Chromosome Identifier (ADCI) software is given. ADCI is developed by a Canadian company (Cytognomix). The papers covering ADCI provide detailed insight into the design of a "pipeline" (detect, then classify) ADS system.

Second, we go over another set of papers written by researchers at LRAcc evaluating DCScore, another proprietary ADS solution designed by MetaSystems. The performance of DCScore in terms of DC detection is insufficient for fully automated use, but it can still be of significant help in a semi-automated setting (where detected DCs are reviewed manually). Finally, we review several recent solutions using deep learning for DC detection.

Chapter 3, describes our contributions at the chromosome level. Three main approaches are described: chromosome classification, chromosome synthesis and training object detection models on synthetic metaphases. In Chapter 4, our proof of concept for

a ConvNet-based ADS system is introduced. Training an effective chromosome classifier is relatively easy with ResNet [8]. However, the performance of this classifier on true chromosomes depends on the distribution gap between real metaphases and our training dataset and on the quality of our chromosome detection algorithm. Building an effective pipeline from this classifier proved to be a difficult task. We did not succeed in building a performant chromosome simulator, but training Faster R-CNN [9] provided insight into the relationship between training dataset size and performance in the context of biological dosimetry.

Chapter 4 introduces our proof of concept for DC detection. This solution uses well-known building blocks like Unet, stochastic first-order optimization, a sparsity-promoting regularizer, and model ensembling. Agregation makes rejection of low confidence detections easy, and relies on interpretable parameters. This ensemble-based approach demonstrates strong performance, both in terms of object detection metrics and in terms of calibration curve estimation.

Scientific communication

The proof of concept for an ADS system was described in an article titled "Ensembling Unets, sparse representation and low dimensional visualization for rare chromosomal aberration detection in light microscopy images" (https://doi.org/10.1101/2023.09.11.557124), which is currently being submitted a journal. This proof of concept was also presented at the *International Conference on Radiation Research* in Montreal (August 2023).

From June to September of 2022, I stayed in the Engineering Physics department at McMaster University, in Canada. This research stay was supervised by Qiyin Fang and focused on assisting Tianqi Hong's PhD work by designing methods for optical flow estimation in microfluidics cell imaging. This optical flow estimation method uses the same sparse regularizer as the proof of concept presented in Chapter 4. This research stay led to a pre-print which is presented in Annex A.

Chapter 1

Deep Learning for image analysis

Deep learning has been at the heart of significant progress in machine learning over the past decade. Image analysis is one of the fields where the influence of Deep Neural Networks (DNNs) has been the most significant. As the application of modern deep learning to chromosomal aberration detection was the driving force behind this PhD project, we present useful references for the rest of the thesis in this chapter.

First, we give a brief introduction of deep learning in the context of image analysis. After introducing Fully Connected Networks (FCNs) and their shortcomings for image data, we give the rationale behind an ubiquitous alternative in image analysis: Convolutional Networks (ConvNets). We introduce backpropagation, and the challenges associated with training Deep Neural Networks (DNNs) using Stochastic Gradient Descent (SGD).

As labelled data is often sparse in biomedical imaging, we provide an introduction to unsupervised deep learning for images. Two main approaches are highlighted: generative models and similarity-based methods.

Providing an accurate modelling of model uncertainty is a key requirement for an automated dosimetry system. Therefore, we provide a review of a few major references in the approximate Bayesian deep learning litterature. In those references, the authors use a Gaussian approximation for the posterior distribution over the weights of the networks. Several ways of estimating the mean and covariance of this Gaussian distribution are provided.

Finally, we give two examples of ConvNet-based object detection models, Faster R-CNN and CenterNet. First, we present Faster R-CNN as an example of a two-stage object detector. We focus on the technical complexity of this model. As an alternative, we highlight the simplicity of one-stage object detectors using CenterNet as an example.

1.1 Deep Learning architectures

1.1.1 ImageNet and deep learning

The ImageNet database [2] was published in 2009 to investigate large scale image classification problems. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) ran yearly, to incentivize and publicize improvements in image classification research. Initially, the best performing methods were popular machine learning techniques at the time, like Support Vector Machines (SVMs) [16]. Those methods usually relied on convex optimization which means that their convergence could be theoretically guaranteed. Before the early 2010s, most methods used hand-crafted features, like Histogram of Gradients (HOG) [17] or Scale Invariant Feature Transforms (SIFT) [18]. In a seminal paper published in 2012, Krizhevsky *et al.* showed that ConvNets trained with SGD brought a considerable improvement on the current performance state-of-the-art [1]. This sparked a renewal of neural networks methods in machine learning, which were previously thought to be very hard to train, owing to the non-convexity of their loss functions.

The development of ConvNets was enabled by the existence of the ImageNet database but also by improvements in computing, both on the software and hardware front. With an appropriate redefinition of the image that trades some memory for speed, convolution can be seen as a matrix multiplication between a kernel and an image, which is an embarassingly parallel problem. The Compute Unified Device Architecture (CUDA) framework made it relatively easy to write fast 2D convolution implementations running on Nvidia Graphical Processing Units (GPUs) with hundreds of cores. While the initial implementation of AlexNet relied on custom CUDA kernels (which required a working knowledge of `C++`), higher level frameworks quickly made deep learning much more accessible by packaging so-called "primitives" (convolutions, Fourier transforms, etc) into simple Application Programming Interfaces (APIs) like PyTorch and Tensorflow [19], [20].

1.1.2 Deep neural networks

DNNs are flexible models composed of stacks of linear and non-linear transformations of the input data. For a two layer neural network, with some activation function σ and an arbitrarily shaped input x, the output of this neural network can be written as:

$$\hat{y} = f_\theta(x) = \sigma(W_2^T \sigma(W_1^T x + b_1) + b_2) \tag{1.1}$$

with $\theta = W_1, W_2, b_1, b_2$ being trainable weights and biases respectively. For $x \in \mathbb{R}^k$, W_1 is a matrix of size $k \times m$ where m is the feature dimension in the first layer, while b_1 is of size m. Here, m is a user-defined parameter. This architecture is arbitrarily flexible: one can add layers, make them larger or smaller or build connections between different layers. The number of successive layers is usually refered as the depth of the neural network. Neural networks are often described as deep, without any agreed-upon standard on the number of layers needed for this description. Because σ is a non-linearity (for example, $\sigma(x) = \frac{1}{1+e^{-x}}$ or $\sigma(x) = \max(0, x)$), extremely complex, non-linear relationships between inputs and outputs can be modelled. In fact, a number of universal approximation theorems exist. A famous example was proven by George Cybenko [21]:

Theorem 1 *Let I_n be the n-dimensional unit cube $[0,1]^n$. The space of continuous functions on I_n is denoted as $C(I_n)$. Let $M(I_n)$ be the space of finite signed regular Borel measures on I_n. We say that σ is discriminatory if for a measure $\mu \in M(I_n)$*

$$\int_{I_n} \sigma(y^T x + \theta) d\mu(x) = 0, \tag{1.2}$$

for all $y \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$ implies that $\mu = 0$. Let σ be any continuous discriminatory function. Then finite sums of the form

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^T x + \theta_j) \tag{1.3}$$

are dense in $C(I_n)$. In other words, given any $f \in C(I_n)$ and $\varepsilon > 0$ there is a sum $G(x)$ for which

$$|G(x) - f(x)| < \varepsilon, \ \forall x \in I_n. \tag{1.4}$$

Training DNNs with first order methods

While Theorem 1 suggest that DNNs are a very flexible class of functions, it tells us nothing about how to approximate a desired function with a neural network *i.e.*, learn the parameters of the model. This is be done by optimizing some loss function $\mathcal{L}$ that depends on data and model parameters. Cross entropy (for a classification problem) or L_2 distance (for a regression problem) are common examples of loss functions. For a set of N input and output $\mathcal{D} = \{(x_1, y_1)...(x_N, y_N)\}$ sampled from some joint probability distribution (X, Y), we can approximate the true risk $\mathbb{E}_{(X,Y)}(\mathcal{L}(X, Y))$ with the empirical

risk, which is also our training criterion:

$$\mathbb{E}_{(X,Y)}\left(\mathcal{L}(X,Y)\right) \simeq \sum_{i=1}^{N} \mathcal{L}(f_\theta(x_i), y_i). \tag{1.5}$$

This empirical risk can be minimized with respect to the model parameters θ, which is called training the model:

$$\theta^\star = \arg\min_\theta \sum_{i=1}^{N} \mathcal{L}(f_\theta(x_i), y_i). \tag{1.6}$$

This simple idea has several issues:

— There is no unique solution to Equation 1.6, and no analytical description of the set of solutions. Therefore, $\theta^\star$ is be estimated using iterative methods that rely on the efficient computation of $\nabla_\theta \sum_i \mathcal{L}\left(f_\theta(x_i), y_i\right)$, as detailed in Section 1.1.2. Note that $\nabla_\theta \mathcal{L}\left(f_\theta(x), y\right) \in \mathbb{R}^k$, with k being the dimension of θ. Therefore, $\mathcal{H}_\theta \mathcal{L}\left(f_\theta(x), y\right) = \nabla_\theta \left[\nabla_\theta \mathcal{L}\left(f_\theta(x), y\right)\right] \in \mathbb{R}^{k\times k}$. As modern neural networks may feature billions of trainable parameters, it is not even be possible to store the Hessian matrix in memory let alone compute it in any reasonable time. Note that despite this fact, a number of fast approximations of the Hessian of DNNs have been suggested, see the papers of Ritter *et al.* [22], [23] or Martens *et al.* [24]. However, the overwhelming majority of deep learning literature still relies on first order methods, like SGD [25] or Adam [12].

— The number of training samples may be extremely large. Internal Google datasets like JFT-3B contain billions of images: computing the gradient of the empirical risk over the complete dataset is not feasible because of memory constraints. Instead, this gradirnt is approximated on a random subset $\mathcal{J}$ of the training data, i.e $\nabla_\theta \sum_{x,y\in\mathcal{D}} \mathcal{L}\left(f_\theta(x), y\right) \simeq \nabla_\theta \sum_{x,y\in\mathcal{J}} \mathcal{L}\left(f_\theta(x), y\right)$, with $|\mathcal{J}| < |\mathcal{D}|$. Thus, model parameters are updated with a noisy estimate of the gradient.

— This training criterion is non-convex; there is no guarantee of finding the global minimum and there is a large number of local minima, usually with comparable loss values. Because of our noisy estimation of the gradient and the fact that we usually do not have any curvature information the exploration of the parameter space is a stochastic process: two successive training runs of the same model with identical initialization but different batch samplings will reach two different parameter values at the end of training. While this was initially seen as a insurmountable issue before

the revival of neural networks in the early 2010s, it has since been suggested that the randomness in parameter space exploration may actually be a key positive factor in deep learning performance. This hypothesis has been named implicit regularization in the litterature, see [26], [27], and is discussed further in Section 1.3.

— Neural networks architecture are extremely flexible: the number of trainable parameters may be similar to the number of training samples, or it may even be much larger. This means that there is a large set of values for θ that achieve a training loss of zero. In fact, surprisingly, DNNs can fit the training data perfectly and generalize very well on unseen test data, see [27]. As explained in the previous item, randomness in batch sampling helps to regularize training. Contrary to previous assumptions, "bad" minimas that do not generalize are rare because of implicit regularization but also thanks to architectural bias embedded in convolutional architectures as we will see in Section 1.1.3.

Computing gradients in network architectures

To train a DNN with first order methods, we need an efficient way to compute $\nabla_\theta \mathcal{L}(f_\theta(x), y)$. Let us consider the same 2-layer FCN as before:

$$f_\theta(x) = \sigma(W_2\sigma(W_1 x + b_1) + b_2), \tag{1.7}$$

with $\theta = \{W_1, W_2, b_1, b_2\}$, where W_1, W_2, and b_1, b_2 denote weights and biases respectively. Let us also define two intermediate quantities at layer l, the output a_l of the activation function σ (called the "activation" of the layer):

$$a_l = \sigma(z_l), \tag{1.8}$$

and z_l an affine transform of the activations at the previous layer, i.e

$$z_l = W_l a_{l-1} + b_l. \tag{1.9}$$

Activations are defined in a recursive fashion: the activations a_l at layer l are a function of the activations at layer $l-1$, composed of an affine transform $W_l a_{l-1} + b_l$ and a non-linear activation function σ applied element-wise. This recursive definition makes representing this class of functions as a computational graph possible, as displayed in Figure 1.1. This

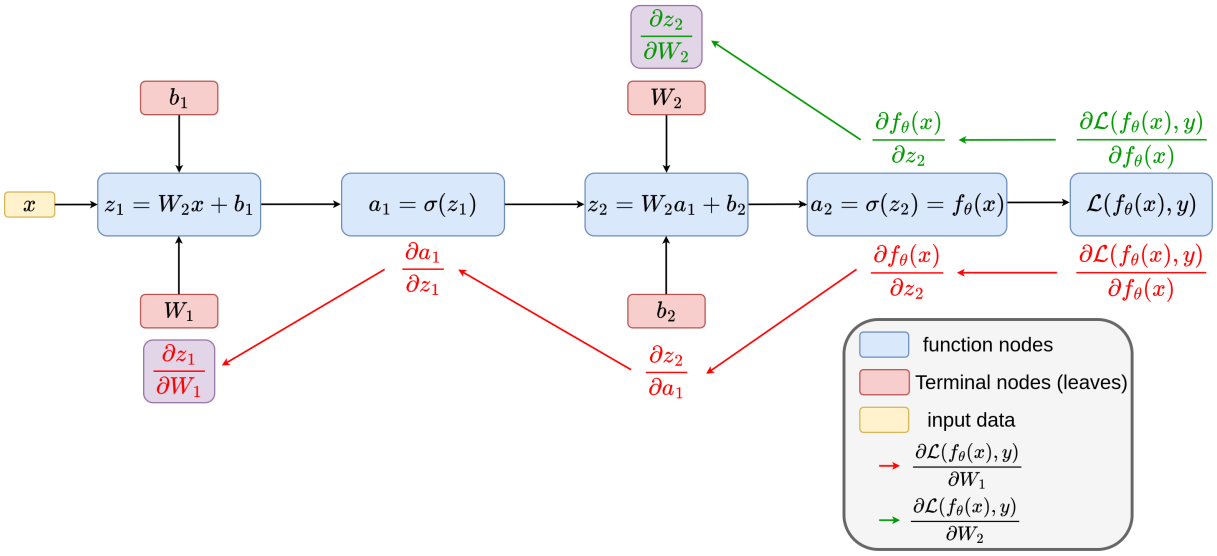


Figure 1.1 – Backpropagation algorithm for a 2 layer neural network. The derivative of every function node (in blue) is computed with respect to the previous one, for all nodes on the path to a leaf (in red). Leaves are trainable parameters.

graph has two types of nodes, either representing function evaluation (in blue) or model parameters (in red) and data (in yellow). Terminal nodes (i.e., nodes that are not used as input for any other node) are called leaves. Usually, leaves are model parameters. Every function node has an associated derivative with respect to its input. During the forward pass, the output of the network with respect to its input is computed. Activations at each function nodes are retained, because they are needed for gradient computation.

During the backward pass, the derivatives of the criterion with respect to trainable parameters are computed for every leaves of the graph. The derivative of this composition of differentiable functions is computed with the chain rule. Derivatives of the loss with respect to a specific leaf are computed by multiplying the derivatives computed at every node along the path to this leaf. For example, the derivative of the loss function with respect to W_2 is

$$\frac{\partial \mathcal{L}(f_\theta(x), y)}{\partial W_2} = \frac{\partial \mathcal{L}(f_\theta(x), y)}{\partial f_\theta(x)} \times \frac{\partial \sigma(z_2)}{\partial z_2} \times \frac{\partial z_2}{\partial W_2}, \tag{1.10}$$

as seen in Figure 1.1. Note that $\frac{\partial z_2}{\partial W_2} = a_1$: the gradient at note i depends on the activations (i.e the output) of the previous node, which is why all activations at every layer are retained during the forward pass. This immensely flexible representation of chained computations is at the core of autodifferentiation, and offers numerous possibilities:

— we can differentiate the loss function with respect to any terminal node (i.e any

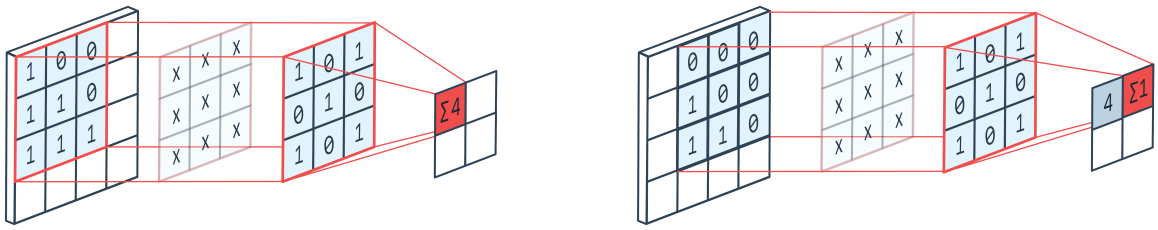


Figure 1.2 – 2D convolution with a (3×3) kernel.

node that is not an input of another node), the difference between input data and a trainable parameter is only semantic. This means that we can differentiate with respect to training data, and even update the input of the model with respect to some criterion. This has been used to find images maximizing the activation of a specific neuron [28], or for style transfer [29] ;

— while neural networks can implement very complex functions, the computation of derivatives is always analytic, without any approximation beyond the limitations of representing real numbers as floats. This is extremely efficient (only one evaluation of the function is needed), and enables the use of lower precision floats (`float32`, or even `float16`) because the additional precision is not as if derivatives were evaluated numerically, which saves memory and computation time.

— In PyTorch, the computational graph is defined lazily: for each sample x_i, a node is added to the graph if and only if a new (differentiable) function is called on the output of a function node of the current computation graph. Nodes can be added and removed based on user-defined conditions, with an automatic adaptation of the backward pass. This means that the computation graph can be different for every sample of the dataset, or modified based on external input. This flexibility is an advantage of lazily defined graphs over compiled graphs, like the one implemented in Tensorflow. However, this flexibility costs some performance.

1.1.3 Convolutional Neural Networks

Fully-connected neural networks architectures do not scale well to images: for an image x of size $H \times W$, a linear transform W_θ of x that does not change the resolution of the input image has $(H \times W)^2$ parameters if we let W_θ be a dense matrix. For $H = W = 256$, W_θ would contain 256^4 `float32`, which is not feasible in practice. Implicitely, for any pixel

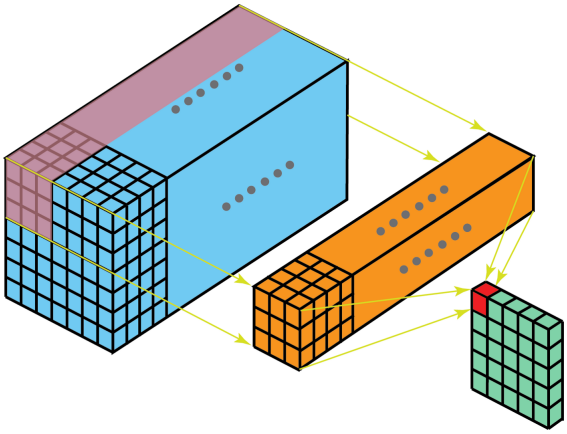


Figure 1.3 – 2D convolution with a multi-channel kernel.

of the image, this large matrix has a trainable coefficient modelling an interaction with any other pixel. ConvNets make two simplifying assumptions:

— pixels usually exhibit high correlation with their closest neighbors but lower correlation with pixels located far away. Instead of keeping the full matrix, long-range interactions can be removed so that only interactions between neighboring pixels are modelled which means that W_θ is block-sparse ;

— a good correlation model with neighboring pixels should be location-independent: instead of learning a local correlation at every location on the image grid, the assumption is that a single correlation model is valid for the whole image ;

In other words, we learn the parameters of a convolution kernel, a set of parameters computing a local average at every location of the image. For any given image, local content interacts with the convolution filter based on their similarity, as shown in Figure 1.2. Given the immense variety of natural images, if one wants to detect some type of object reliably, a single convolutional filter is not enough: a filter providing a strong response for some orientation and lighting condition might not be activated reliably by the same object in other conditions. Furthermore, objects in natural images exhibit scale variation, so that a filter might be activated by the object if it is sufficiently distant, but it might not provide useful features if the object is closer. For those reasons, ConvNets rely on a large number of convolution filters, stacked in successive "blocks", to provide an expressive and hierarchical representation of the image, as shown in Figure 1.3. Obviously, convolution did not appear in the deep learning litterature first, and has been a core component of image analysis for decades. For example, horizontal and vertical gradients can be computed using convolution filters, as shown in Figure 1.4.

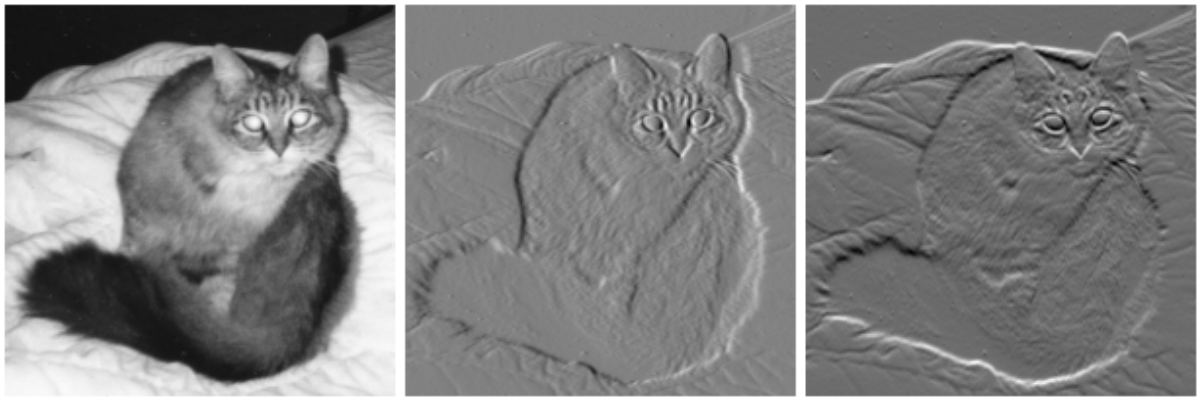


Figure 1.4 – Left: input image, middle: horizontal gradient, right: vertical gradient.

Efficient implementation of 2D convolution

A naive implementation of convolution might rely on two `for` loops, iterating over the height and width of the input image and multiplying the local intensities of the image with the convolution kernel. However, this fundamental operation can be parallelized very effectively, as described in this section.

Let us define a batch of $\mathcal{J}$ images, with spatial dimensions (H, W) and N_i channels. For some convolution kernel with spatial size K_H, K_W, a stride S (the step size of the convolution kernel) and a padding of P pixels, the output of the convolution operation has the following spatial size:

$$H_o = \left(\frac{H - K_H + 2P}{S}\right) + 1,\ W_o = \left(\frac{W - K_W + 2P}{S}\right) + 1. \tag{1.11}$$

Using this, the batch of images image can be divided into N_p overlapping patches in an unfolded view, which is an array with dimensions $[\mathcal{J}, K_H \times K_W \times N_i, N_p]$. Here, $N_p = H_o \times W_o$ with H_o and W_o being the spatial size of the convolutions output. Note that this representation is redundant because patches are overlapping: some pixels are represented multiple times. This means the unfolded representation takes up more memory than the initial, "folded" representation.

For a 2D convolution taking an image with N_i channels as input and outputting another image with N_o channels, the convolution kernel (i.e the trainable parameters) is an array of size $[N_o, N_i, K_H, K_W]$. This convolution kernel can be flattened to an array of size $[N_o, N_i \times K_H \times K_W]$, so that the unfolded batch of size $[\mathcal{J}, K_H \times K_W \times N_i, N_p]$ and the flattened convolution kernel can be multiplied together. The output batch is of size

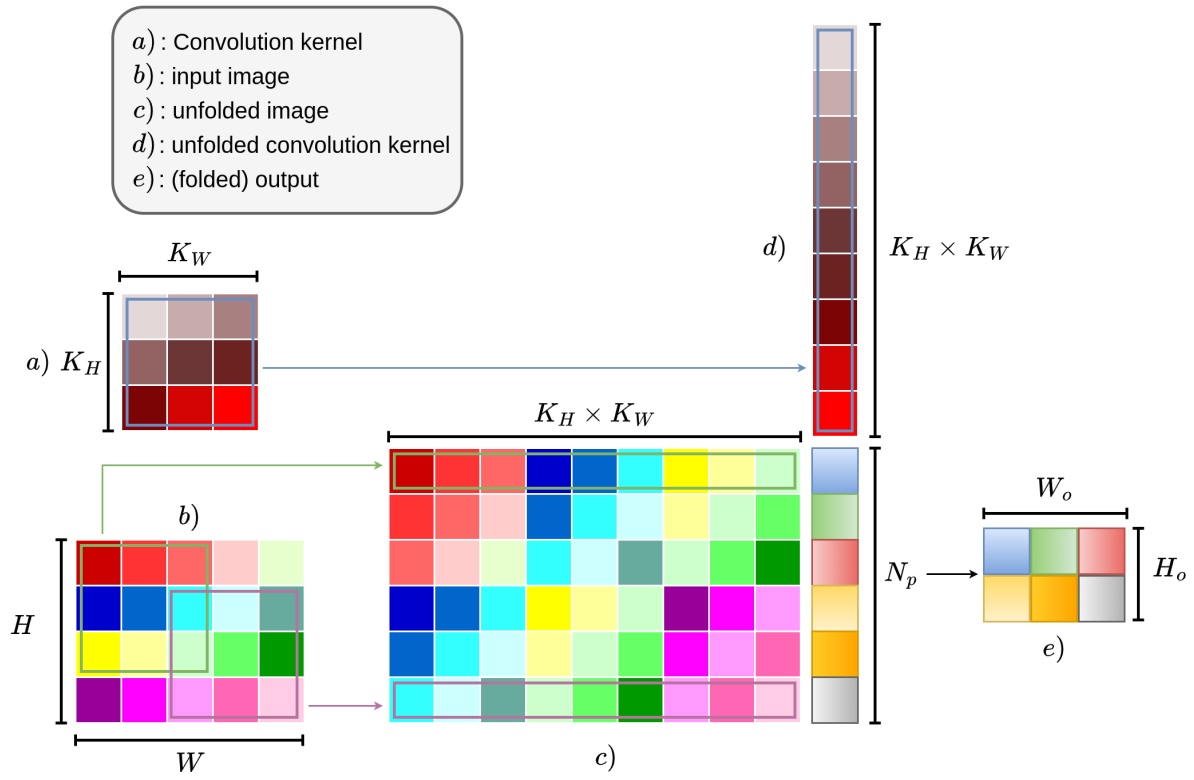


Figure 1.5 – Unfold-GEMM-Fold implementation of convolution showcased on a 1 channel image.

$[\mathcal{J}, N_o, N_p]$, i.e $[\mathcal{J}, N_o, H_o \times W_o]$, which can be reshaped into $[\mathcal{J}, N_o, H_o, W_o]$.

The local averaging effect of the convolution can clearly be seen in the output shape: the $K_H \times K_W$ term has disappeared, as the output at every spatial location of the input corresponds to the sum of the element-wise product between the patch and the kernel. Furthermore, this representation makes the fact that convolution is a linear operation obvious. In turns, this explains why computing derivatives in ConvNets is easy, as they are mostly stack of linear transforms and element-wise non-linearities.

As shown in Figure 1.5, 2D convolution can be implemented with General Matrix Multiply (GEMM) , which makes parallel implementations easy. Because matrix multiplication is an embarassingly parallel problem, this makes accelerating this computation of GPUs very easy, but increases memory consumption. This increase is the largest limitation in scaling up convolutional models.

Architectural bias

Every ConvNet is a FCN, but even for small images where FCNs might be feasible, ConvNets are usually much easier to train. This is because the assumptions made in section 1.1.3 introduce architectural bias which makes modelling images easier. To use an alternate wording, the statistical learning community would say that ConvNets are a good prior model to describe images. This bias has been extensively studied, and has led to a number of surprising discoveries:

— Completely random ConvNets can be effective image priors for image restoration tasks, as shown in Deep Image prior [30]. For some noise vector z and a corrupted image x, finding $\theta^\star = \arg\min_\theta \|f_\theta(x) - z\|_2^2$ with early stopping yields good performance on inverse problems, even though the model is trained to reconstruct noise;

— Training only the batch normalization parameters, or only a very small subset of the model parameters is sufficient to reach non-trivial levels of performance, as shown in [31], [32];

— ConvNets can be used as a prior for a FCN. In [33], d'Ascoli *et al.* find that the weights of a trained convolutional neural network can be used as an initialization for a FCN. FCNs initialized that way reach a higher level of performance than purely convolutional architectures.

Receptive field

In most cases, convolutional architectures use downsampling to agregate information from different spatial locations in the image, either by using large convolutional kernels without padding, or by pooling pixels, retaining only the maximum or the average of the sub-region of the image. This spatial agregation means that as we reach deeper layers of the network, a single pixel in feature space agregates information from a large subset of the input in image space.

More formally, for a fully convolutional neural network f_θ, a_l is the output activation volume at layer l. As we are in a convolutional architecture, we have $a_l \in \mathbb{R}^{H/K \times W/K \times N_i}$ with W, H the input image height and width, K some downsampling factor that depends on the stride and padding of the successive convolutions and N_i a pre-defined number of convolution kernels (also called channels). For a given feature pixel $a_l[u, v]$ for $u, v \in \{0...H/K - 1\} \times \{0...W/K - 1\}$ (corresponding to a vector of size N_i), the receptive

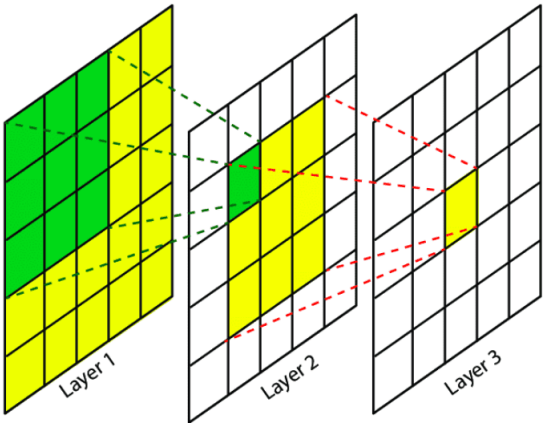


Figure 1.6 – Receptive field in ConvNets.

field of this pixel corresponds to the specific subset of the input image $x[U, V]$ (with U, V a set of image coordinates) so that $a_l[u, v] = f_\theta(x[U, V])$, as shown in Figure 1.6. There is a tradeoff between the receptive field and spatial resolution of a ConvNet: as the receptive field grows larger and the model agregates information from distant locations of the input image, the spatial resolution is reduced: a single pixel models a large part of the input image. Therefore, accurately modelling objects at different scales in the same image requires specific architectural choices.

1.2 Unsupervised learning for image analysis

The early successes of deep learning depended on the availability of large annotated datasets like ImageNet [2], COCO (Common Objects in Context) [3] or Pascal VOC [4]. However, curating large annotated datasets is expensive, especially when it requires specialized knowledge, which is extremely common in biomedical imaging. While the deep learning community quickly noticed that pre-training on ImageNet was usually an effective way to achieve good results with limited amounts of data this solution is less effective when there is a large domain gap between ImageNet and the application domain like in satellite or biomedical imaging. Many imaging modalities found in those applications are not even available in ImageNet, like hyperspectral or volumetric imaging.

Therefore, unsupervised training methods that do not rely on annotated data are needed. Those training techniques should learn useful features from an unlabelled dataset $\mathcal{D} = \{x_1, \cdots, x_N\}$. In other words, a pretext task is needed, which can be solved without

any labels. This pretext task is solved by training a feature extractor which can then be reused for supervised downstream tasks. Designing effective pretext tasks is a longstanding challenge, as different downstream tasks may requires different features. In this section, we focus on two research directions for unsupervised learning in image analysis: generative models, and similarity-based approaches.

A generative model is trained by learning a data distribution from a set of samples. The curse of dimensionality makes it difficult for images, but a number of successful methods have been proposed, like Generative Adversarial Networks (GANs) or Variational Autoencoders (VAEs).

In similarity-based approaches, a similarity criterion is optimized between different augmented views of the same image. Augmentations can be noise, blurring, flipping, color shifting ... This way, the network learns features that are invariant to those perturbations. The minimization of those criterions usually have trivial solutions, where the same features are predicted for all images, which is called representation collapse. Preventing this collapse is a central issue in similarity-based methods.

This section is not intended as an exhaustive survey of unsupervised feature learning for images as thousands of pages could be written on the subject, even by restricting ourselves to the deep learning literature. We focus on a few impactful architectures to highlight the similarity, differences and tradeoffs between those training criterions.

1.2.1 Generative models

A number of approaches have tried approaching the unsupervised learning problem as a distribution learning problem. As images are very high dimensional, non-parametric distribution learning techniques suffer from the curse of dimensionality: sample density is extremely low in image space [34]. In the deep learning era, two noteworthy approaches are GANs [35] and VAEs [36]. While adversarial models have managed to produce impressive samples [37], the adversarial criterion remains difficult to optimize [38]. On the other hand, VAEs enjoy a more principled training criterion but fail to match GANs on sample quality.

Generative Adversarial Networks

GANs learn to transform low dimensional Gaussian samples from a latent distribution $p_z(z)$ into images with an adversarial training scheme between a generator G and a dis-

criminator D. The discriminator tries to predict wether or not any given image is sampled from the learned generator distribution $p_g(x)$ or from the true data distribution $p_r(x)$.

In a seminal paper, Goodfellow *et al.* [35] showed that generating images using this aforementioned adversarial game was possible. A generator function $G(z, \theta) : \mathbb{R}^l \to \mathbb{R}^d$ maps Gaussian samples to data space, with $l < d$. A discriminator $D(x, \phi) : \mathbb{R}^d - > \mathbb{R}$ solves a binary classification task: it predicts wether the input is a sample of the generator distribution $p_g(x)$ or the real data distribution $p(x)$. Therefore, the discriminator outputs a single scalar.

The generator G is trained to minimize $\log(1 - D(G(z)))$, *i.e.* the probability that the discriminator correctly identifies the output of the generator as fake. On the other hand, the discriminator is trained to maximize the probability of correctly classifying real and fake samples. The training objective of a standard GAN, as proposed in Goodfellow *et al.* is:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_r(x)} \left[\log D(x)\right] + \mathbb{E}_{z \sim p_z(z)} \left[\log(1 - D(G(z)))\right]. \tag{1.12}$$

The min-max formulation makes optimizing this criterion difficult. Optimizing the discriminator to convergence before updating the discriminator would be too costly, so most GAN training schemes alternate updates of the generator and discriminator. Training can also collapse rather easily, especially in early phases of training. When the generator provides poor samples, classifying generator output is easy, so that $D(G(z))$ is very close to 0. The generator gradient update (where m is the batch size)

$$\nabla_\phi \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D(G(z_i))\right), \tag{1.13}$$

is very small. To prevent this, Goodfellow *et al.* suggest maximizing $D(G(z))$ instead, but collapse of training because the discriminator is too strong remains an issue in GAN training. In fact, [38] proved that the gradient updates of the generator tend to 0 as the discriminator improves, under some conditions on $p_g(x)$ and $p_r(x)$. In [39], the authors replace the FCNs used in [40], [41] with ConvNets, which improves training stability and performance. Furthermore, they show that the discriminator of a GAN can be used as a feature extractor to solve downstream task like image classification. Additional study of GAN training stability was provided in [42]. In Mirza *et al.* [41], the training criterion of (1.12) is rewritten for class-conditional generation:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_r(x)} \left[\log D(x, y)\right] + \mathbb{E}_{z \sim p_z(z)} \left[\log(1 - D(G(z, y)))\right], \tag{1.14}$$

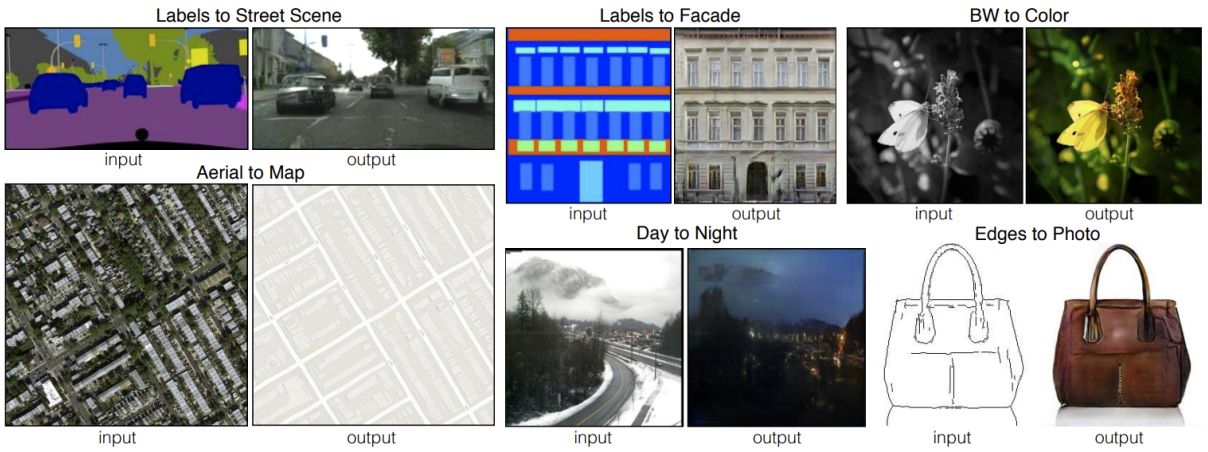


Figure 1.7 – Image-to-image translations capabilities of pix2pix [43].

where y denotes labels. While initially, Mirza *et al.* showed that GANs can be conditionned with class information, more sophisticated conditioned conditional GANs were later demonstrated. For example, in [43] Isola *et al.* show that GANs can be conditioned with images, for image-to-image translation, as shown in Figure 1.7. This architecture is called pix2pix. The authors use the following criterion:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_r(x)}\left[\log D(x,y)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1-D(G(z,y)))\right] + \mathbb{E}_{x,y,z}\left[\|x-G(z,y)\|_1\right]. \tag{1.15}$$

The $\mathbb{E}_{x,y,z}\left[\|x-G(z,y)\|_1\right]$ term improves sample quality by forcing the generator to produce outputs that match the true input image. The generator G is conditioned both on the noise z and the label y. However, the authors note that there is little stochasticity in the decoder output: sampling new noise vectors does not lead to a diverse output of the generator. Variants of pix2pix where the generator output does not depend on any noise produce almost identical results to the model trained with the criterion in (1.15).

Deterministic and probabilistic autoencoders

This section introduces autoencoders, first as a compression model and then as a latent variable model of a true data distribution, as introduced in the famous paper by Kingma *et al.* [36]. The similarities and differences between the two approaches are highlighted, as presented in [44].

Autoencoders rely on a simple idea: learning how to compress an image, and reconstruct it from a low-dimensional latent representation requires learning good image fea-

tures. Image compression was frequently used as a pretext task in early deep learning experiments [45]. This class of models can be described in the following way: an input image x, with dimension d is compressed to a latent representation $z \in \mathbb{R}^k$, $k < d$ by a parameterized function $f_\phi(x)$. This latent representation is decompressed into a reconstruction of the input image $\hat{x} = g_\theta(z) = g_\theta(f_\phi(x))$. The parameters (θ, ϕ) can be learned by minimizing a reconstruction error, for example the squared euclidean norm:

$$\theta^\star, \phi^\star = \arg\min_{\phi,\theta} \|x - g_\theta(f_\phi(x))\|_2^2. \tag{1.16}$$

If f_ϕ and g_θ are defined as linear transforms and the reconstruction error is the L_2 norm, the optimal encoder and decoder correspond to the first k vectors of the Principal Component Analysis (PCA) basis. Using DNNs, deep autoencoders can be trained with backpropagation and first order optimization. Kingma *et al.* refined deep autoencoders further in [36], by parameterizing a probability model for a random variable x with DNNs. In this paper, the authors suggest modelling x as a function of a latent unobserved variable z:

$$X \sim p_\theta(x|z) = p(x|g_\theta(z)), \tag{1.17}$$

where g_θ is a deterministic function mapping the latent space to the data space, usually called a decoder. In the same way, the probability distribution over the latent space is defined as

$$z \sim q_\phi(z|x) = q(z|f_\phi(x)), \tag{1.18}$$

where f_ϕ is a deterministic function mapping data space to latent space, usually called an encoder. The likelihood (or model evidence) of this model is

$$p(x) = \int p(x, z)dz, \tag{1.19}$$

which requires evaluating the joint distribution over the whole dataset. This integral is usually intractable and an alternative formulation is needed. One can show that the log-evidence can be decomposed as

$$\log p(x) = \mathrm{ELBO}(\theta, \phi) + \mathbb{KL}\left[q_\phi(z|x)|p(z|x)\right], \tag{1.20}$$

where $p(z|x)$ is the true model posterior, defined as

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}, \tag{1.21}$$

which again depends on the intractable model evidence. The Evidence Lower BOund (ELBO) can be rewritten as

$$\text{ELBO}(\theta, \phi) = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x|z)\right] - \mathbb{KL} \left[q_\phi(z|x)|p(z)\right], \tag{1.22}$$

where $p(z)$ is a prior on the latent space. Because of the Jensen inequality, the Kullback-Leibler (KL) divergence between the true and parametric posterior is always positive, and the model evidence $p(x)$ is a constant. Therefore, maximizing the ELBO with respect to the parameters ϕ, θ is equivalent to minimizing the KL divergence between the true and parametric posterior. To optimize the lower bound of (1.22), specific probability models for q_ϕ and p_θ are needed. Usually, in the context of VAEs, Gaussian probability models are chosen:

$$q_\phi(z|x) = \mathcal{N}\left(\mu_\phi(x), \sigma_\phi(x)\right), \; p_\theta(x|z) = \mathcal{N}\left(\mu_\theta(z), (\sigma_\theta(z)\right), \tag{1.23}$$

where μ_ϕ, σ_ϕ are functions outputting a mean and a variance for the latent probability model, so that $f_\phi(x) = (\mu_\phi(x), \mu_\sigma(x))$ and $g_\theta(z) = (\mu_\theta(z), \sigma_\theta(z))$. The first expectation term in the ELBO must be estimated with samples of the latent distribution. For computational reasons, a single sample is usually used, which makes the decoder effectively deterministic. Therefore, we have

$$\mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(X|z)\right] = \|x - \mu_\theta(\mu_\phi(x))\|_2^2. \tag{1.24}$$

Let us focus more closely on z. Equation 1.23 tells us that the latent probability model is a Gaussian variable. Therefore, we have

$$p_\phi(z|x) = \mu_\phi(x) + \sigma_\phi(x) \odot \varepsilon, \; \varepsilon \sim \mathcal{N}(0, I), \tag{1.25}$$

Where $\odot$ is point-wise multiplication (Hadamard product). Therefore, as pointed out by Ghosh *et al.* in [44], in practice, VAEs are noise-regularized autoencoders. The deterministic latent representation is corrupted with Gaussian noise, and the variance of this noise depends on the input image. Finally, the second term of (1.22) is a KL divergence between

two Gaussian distributions, which can be computed analytically:

$$\mathbb{KL}(q_\phi(z|x)|p(z)) = \frac{1}{2}\left(\|\mu_\theta(x)\|_2^2 + d + \sum_{i=1}^{d}(\sigma_\phi(x)_i - \log\sigma_\phi(x))\right). \tag{1.26}$$

This divergence is minimized if the trained encoder outputs a centered Gaussian with unit variance. This is a regularization term that tends to smooth the latent space of the autoencoder.

In the initial definition of the ELBO, the reconstruction term of (1.24) and the regularization term of (1.26) are equally balanced. However, in practice, training this model can be difficult. Often, some kind of annealing schedule is needed, where the training starts without any regularization, and progressively increases the weight of the KL divergence term of (1.26) in the ELBO. The training criterion of (1.22) can be rewritten to depend on an hyperparameter β which balances reconstruction and regularization:

$$ELBO(\theta, \phi, \beta) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] - \beta\mathbb{KL}\left[q_\phi(z|x)|p(z)\right]. \tag{1.27}$$

Here, β can be increased beyond 1, which over-regularizes the model, trading off reconstruction performance for a smoother latent space. A surprising effect of this over-regularization is that it tends to produce disentangled latent spaces, as shown in [46].

In a disentangled latent space, every dimension of the latent space controls a specific type of variation in output space. This property has not received a rigourous definition, because defining semantic variations identified by humans in images is close to impossible. However, visual examples can provide a better intuition of this property, as shown in Figure 1.8.

Regularized autoencoders

While VAEs provide a complete probability model for images, where traditional density estimation methods (like kernel density estimation) usually fail, practical implementations of this probability model are usually fairly simplistic. Noticing that VAEs are effectively noise-regularized autoencoders, Ghosh *et al.* compare VAEs to Regularized Autoencoders (RAEs) in [44]. The following criterion is optimized:

$$\begin{aligned}\mathcal{L}_{RAE} &= \mathcal{L}_{REC} + \beta\mathcal{L}_Z^{RAE} + \lambda\mathcal{L}_{REG} \\ &= \|x - g_\theta(f_\phi(x))\|_2^2 + \beta\|f_\phi(x)\|_2^2 + \lambda\mathcal{L}_{REG}.\end{aligned}$$

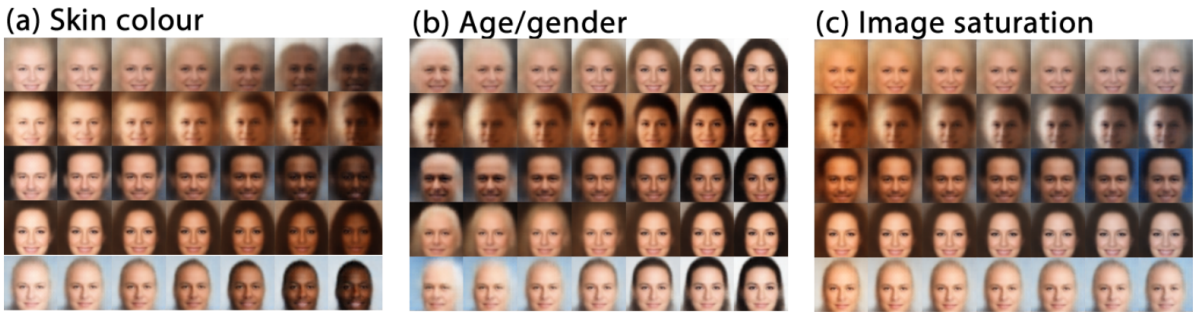


Figure 1.8 – Latent factors learned by β-VAE on celebA: specific dimensions of the latent space capture factors of semantic variations in images. Note that some semantic factors are not disentangled, like age and gender in b). (*Source*: [46])

We re-use the deterministic autoencoder notations introduced in Section 1.2.1. The $\beta\mathcal{L}_Z^{RAE}$ is used to constrain the latent variable near the origin of the latent space, to prevent uncontrolled "growth" of the latent space which could lead to numerical issues.

The decoder g_θ is architecturally close to a GAN generator, as it transforms a latent representation into an image. Therefore, common GAN generator regularizer can be used. Tikhonov regularization on the decoder weights is equivalent to weight decay, which is commonly used. Two noteworthy examples for GAN generators are gradient penalty regularization, $\mathcal{L}_{REG} = \|\nabla_{\phi,\theta} g_\theta(f_\phi(x))\|$[42], and spectral normalization [47], where weight matrices are divided by an estimate of their largest singular values.

Finally, the authors of [44] note that this deterministic formulation prevents any sampling from the latent space, which is a major advantage of VAEs. To remedy this, they fit a Gaussian mixture model on the latent space once model training is finished. For a random sample z_{new} of this mixture model, $g_\theta(z_{new})$ provides a decoded view.

1.2.2 Similarity-based methods

While generative pretext tasks can be easy to train (although some of them are fairly unstable, like GANs), there is no guarantee that the features learned during training will be useful for discriminative downstream tasks like classification. Therefore, pretext tasks that are better suited to discriminative tasks have been designed. In this section, we review a broad class of methods that can be summarized as "similarity based": some kind of similarity metric between augmented views of the same batch of images is optimized during training.

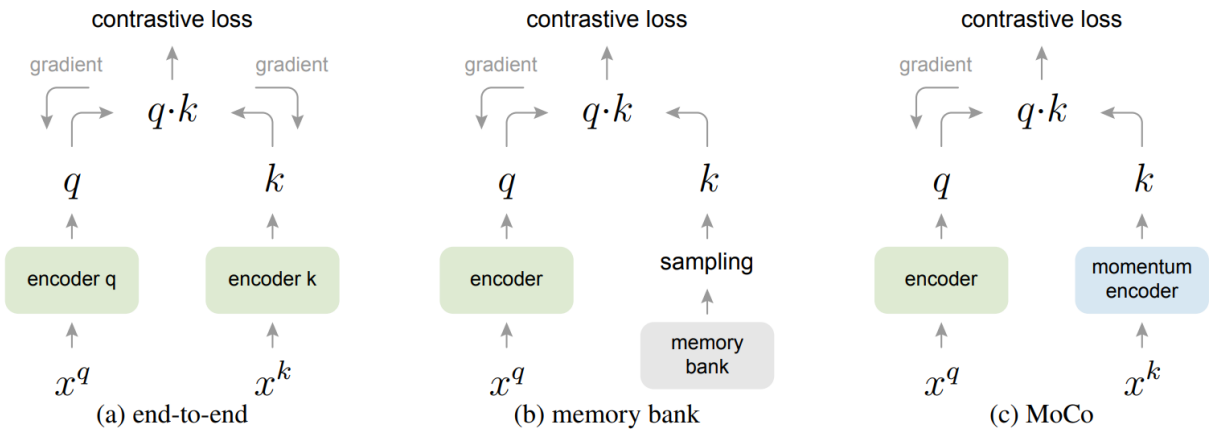


Figure 1.9 – Conceptual comparison of three contrastive loss mechanisms (empirical comparisons are in Figure 3 and Table 3). Here we illustrate one pair of query and key. The three mechanisms differ in how the keys are maintained and how the key encoder is updated. (a): The encoders for computing the query and key representations are updated end-to-end by back-propagation (the two encoders can be different). (b): The key representations are sampled from a memory bank [50]. (c): MoCo encodes the new keys on-the-fly by a momentum-updated encoder, and maintains a queue (not illustrated in this figure) of keys. (*Source*: [48])

Contrastive learning

In contrastive learning, the pretext task is discriminative: the model is trained by minimizing a classification loss. For a batch of images, two views are produced with random augmentations for each image. The loss is computed over all pairs of images. The model should attribute high probability to pairs of matching views, and low probability to other pairs of images. In all contrastive approaches, selection of negative samples is a major topic: if the set of negative samples is extremely small, trivial features might be enough to solve the classification problem. On the other hand, enlarging the set of negative samples brings computational challenges. We review two approaches in detail: simCLR (Simple Contrastive Learning of Representations) and MoCo (Momentum Contrast) [48], [49].

In MoCo [48], the authors frame the contrastive learning problem as a dictionnary lookup task, which can be used as a general framework to highlight the difference between the approaches suggested in the litterature. For any image $x \in \mathcal{D} = \{x_1, \cdots, x_N\}$, a query encoder f_q produces a query $q \in \mathbb{R}^d$. This query is matched to a set of keys $\{k_1, \cdots, k_m\}$. The dictionary contains a single positive key k_+ (which corresponds to another augmented

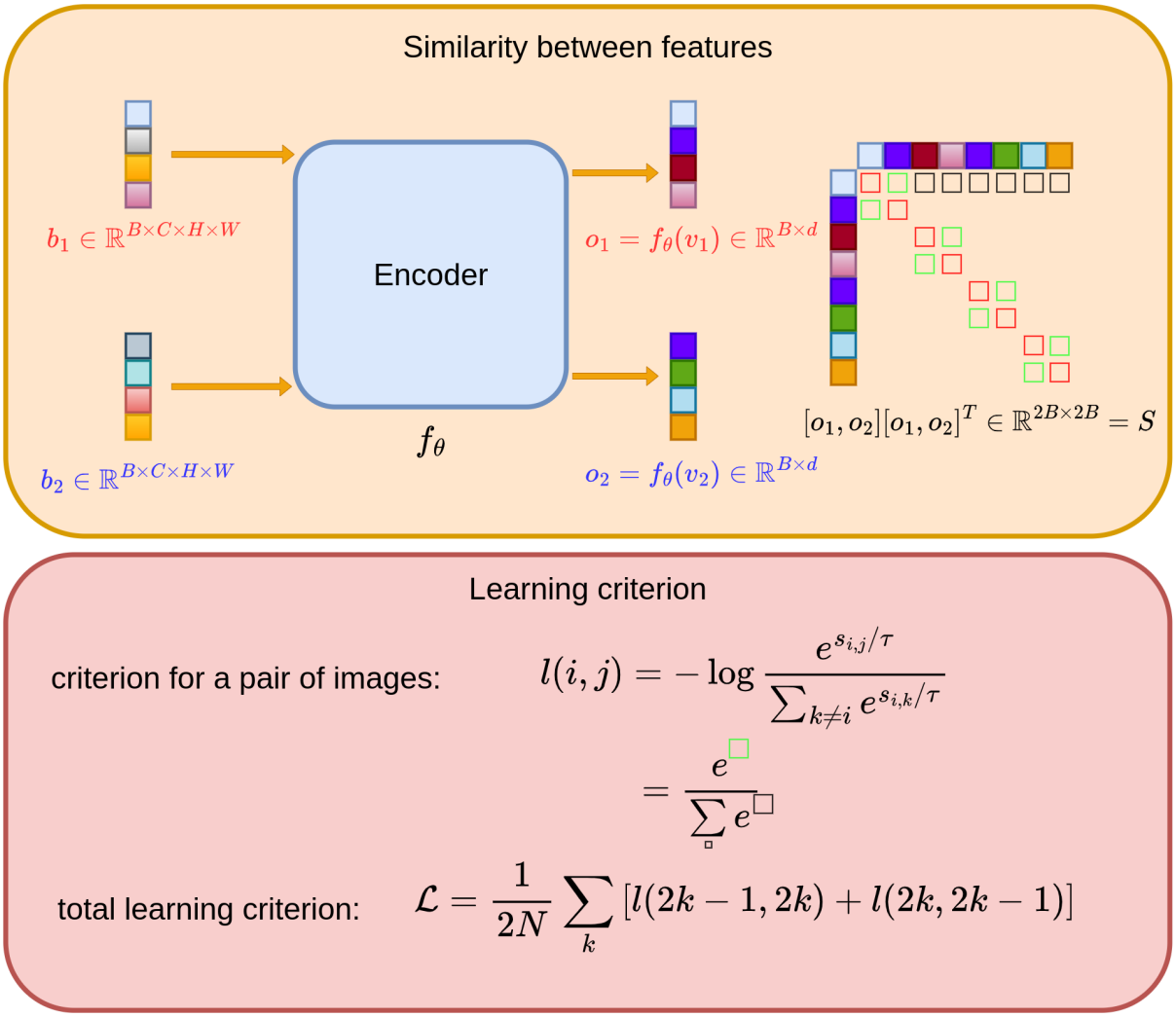


Figure 1.10 – In simCLR, f_θ is trained to maximize similarity between different views of every key in the batch (i.e matching keys), and to minimize similarity between non-matching keys

view of the same image), while all the other negative keys k_- correspond to images $x_j, j \neq i$. for a single key (i.e a single image), the training criterion is:

$$\mathcal{L}_q = \frac{e^{q\cdot k_+}}{\sum_{k_-} e^{q\cdot k_-}}. \tag{1.28}$$

This way, the similarity between views of the same image is maximized, while the similarity between different images is minimized. The fact that the model simultaneously maximize similarity between positive pairs and dissimilarity between negative pairs is the

key ingredient to prevent representation collapse. Contrastive methods for unsupervised learning mostly differ in how the key encoder and the set of negative keys are defined, as explained in Figure 1.2.2. In MoCo [48], the query encoder is a moving average of the key encoder (with the same architecture), i.e

$$\theta_k = m\theta_k + (1-m)\theta_q, \tag{1.29}$$

with θ_k, θ_q being respectively the parameters of the key and query encoder. In SimCLR [49], The key and query encoder are shared, so that $\theta_k = \theta_q$. In MoCo, the set of keys is continuously updated, but kept much larger than a single batch to avoid making the discriminative task trivial. The set of keys is a queue: the oldest keys are removed at every batch step, while the new batch represents the newest member of the dictionnary.

SimCLR [49] avoids queue mechanisms by using a very large batch sizes. This leads to a very simple architecture, but requires very large batches (around 4k images on ImageNet), which quickly becomes intractable with large images (like in biomedical imaging). In this architecture, there is only one single shared model that is trained using a contrastive loss. There is no dictionary, and for a positive key, the matching key corresponds to an augmented view of the same image.

Clustering-based approaches

In supervised image classification, we have a class label y_i for every image x_i. Even if this label is not available, we could still optimize a classification loss if we had some proxy label that corresponds to semantic characteristics but does not rely on human annotations. This pseudo-label should be retrieved in an automatic fashion, which is hard because this is effectively equivalent to the supervised classification problem.

For datasets like MNIST featuring well-defined factors of variation [51], simple feature learning methods like PCA might be enough to build relatively well-defined clusters based on pixel values in an unsupervised fashion. However, this is not possible for rich and complex datasets like ImageNet [2].

In [52], Caron *et al.* proposed an iterative strategy that relies on bootstrapped labels and circumvents the need for human annotations. Initially, image embeddings are computed using an untrained ConvNet. Then, pseudo-labels are built from cluster assignments computed with K-Means. Finally, the ConvNet is trained with a conventional cross-entropy loss using those pseudo-labels. After every training epoch, cluster assign-

ments are re-computed, using the embeddings provided by the classification network.

To prevent a trivial solution where all images are mapped to a single cluster, the centroid of an empty cluster is recomputed from the centroid of a non-empty cluster with an additional random perturbation, and K-Means assignments are re-computed. This effectively splits the non-empty cluster into two. Furthermore, to prevent the model from predicting a single representation, the classification loss of a single sample is weighted based on the number of images in the cluster, i.e, if we have K different clusters of sizes $\{n_1, \cdots, n_K\}$

$$\mathcal{L}(X, Y) = \sum_i \lambda_i y_i \log s(f_\theta(x_i)), \tag{1.30}$$

with $\lambda_i = \frac{1}{n_p}$ if sample i belongs to cluster p. Therefore, high classification performance on large clusters brings a lower reduction in loss, which prevents representation collapse.

While this approach is effective, cluster assignments are computed offline, which means that it does not scale very effectively to large datasets. After every epoch of training, clusters have to be re-computed with K-Means. In [53], Caron *et al.* optimize a similar classification loss based on pseudo-labels. However, those pseudo labels are computed in a different fashion: instead of computing assignments using K-Means on feature vectors, assignments are computed using the Sinkhorn-Knopp algorithm [54], as explained below.

A set of B feature vectors $Z = \{z_1, \cdots, z_B\}$ needs to be mapped to a set of K prototypes $C = \{c_1, \cdots, c_K\}$. Feature vectors should be mapped to prototypes on the basis of their similarity. This mapping $Q \in \mathbb{R}_+^{K \times B}$ is the solution of the following optimization problem:

$$\arg\max_{Q \in \mathcal{Q}} Tr(Q^T C^T Z) + \varepsilon H(Q) \tag{1.31}$$

with $H(Q) = \sum_{i,j} Q_{ij} \log Q_{ij}$, ε taking a low value ($\varepsilon \simeq 0.001$), and

$$\mathcal{Q} = \left\{ Q \in \mathbb{R}_+^{K \times B},\ Q 1_B = \frac{1}{K} 1_K,\ Q^T 1_K = \frac{1}{B} 1_B \right\}. \tag{1.32}$$

The constraints in (1.32) prevent solutions where all images are mapped to the same prototype, by ensuring that the distribution of feature vectors over the different prototypes is roughly uniform.

The rest of the paper uses conventional ideas in unsupervised learning. We have two augmented views x_s, x_t of the same batch of images. A ConvNet f_θ builds two sets of features $z_s = f_\theta(x_s), z_t = f_\theta(z_s)$ from those augmented views. For each view, those features

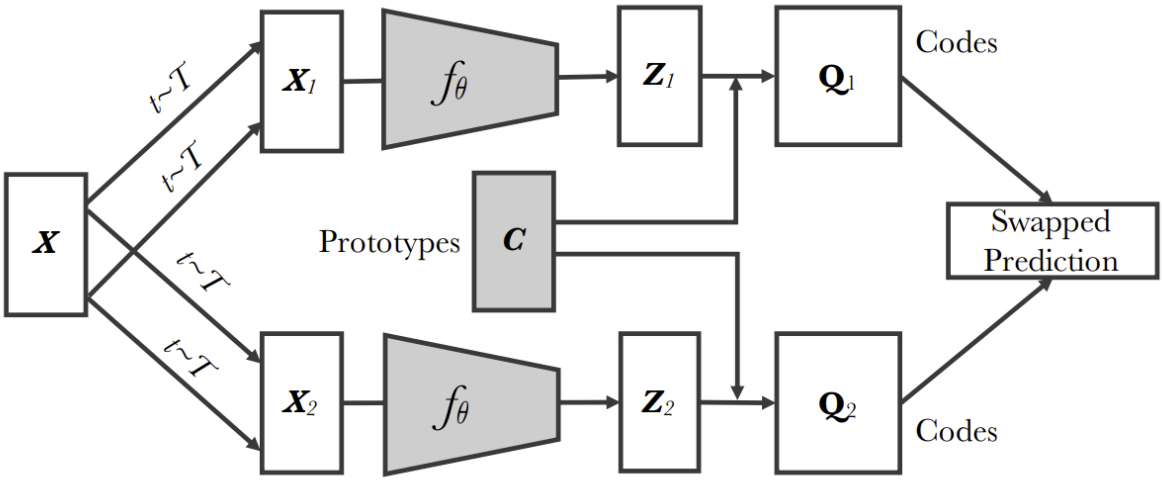


Figure 1.11 – Swapping Assignments between Views (SWaV). (*Source*: [53])

are mapped to prototypes using the Sinkhorn-Knopp algorithm, yielding sets of codes q_s, q_t. Finally, the ConvNet f_θ is trained with the following criterion:

$$\mathcal{L}(z_s, z_t) = -\left(\sum_k q_s^k \log p_t^k + \sum_k q_t^k \log p_s^k\right) \tag{1.33}$$

with

$$p_s^k = \frac{\exp\left(\frac{1}{\tau} z_s^T c_k\right)}{\sum_{k'} \exp\left(\frac{1}{\tau} z_s^T c_{k'}\right)}. \tag{1.34}$$

The model is trained by predicting the prototype assignments of one view from the features of the other views. This ensures that f_θ produces features that are invariant to the augmentations used to produce the views x_s, x_t. This process is summed up in Figure 1.11.

Similarity-maximization approaches

In [55], Chen *et al.* point out that a number of successful unsupervised learning architectures that avoid representational collapse have been designed (see Sections 1.2.2, 1.2.2). However, there has been little interest in the simplest architecture that can avoid this issue. For example, one could ask if negative samples are absolutely necessary to prevent representation collapse. Chen *et al.* suggest that this is not the case, and an architecture

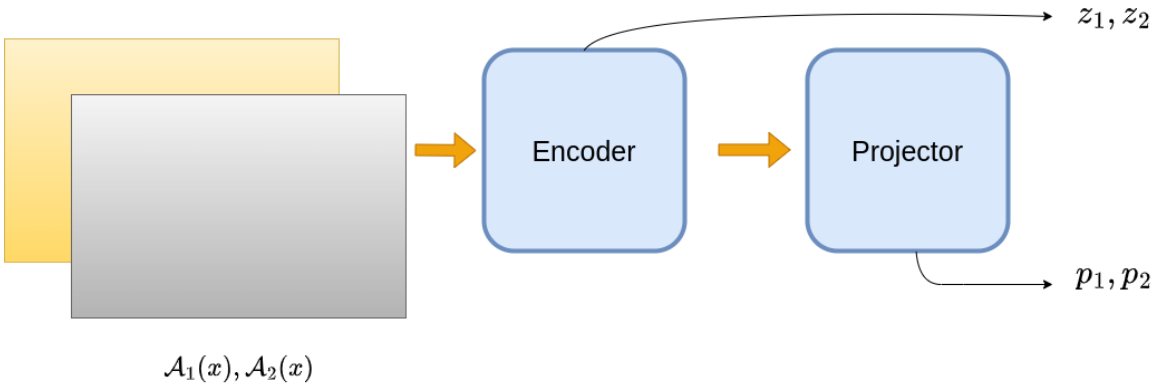


Figure 1.12 – SimSiam architecture. Note that the dimension of z and p are identical.

that relies only on maximizing similarity between positive samples is feasible. SimSiam [55] relies on a encoder network $f_\theta(x) : [0,1]^{H\times W\times C} \to \mathbb{R}^d$ and a projector $p_\theta(z) : \mathbb{R}^d \to \mathbb{R}^d$, as shown in Figure 1.12. This neural network is trained using the following criterion:

$$\mathcal{L} = \frac{1}{2}D(p_1, \text{sg}(z_2)) + \frac{1}{2}D(p_2, \text{sg}(z_1)), \tag{1.35}$$

where sg denotes the stop-gradient operator. The similarity criterion is cosine similarity, i.e.,

$$D(u, v) = \frac{u^T v}{\|u\|_2 \|v\|_2}. \tag{1.36}$$

The key ingredient to prevent representation collapse is the stop-gradient operator. This means although $z = f_\theta(x)$, no gradient is computed with respect to θ for this term of the loss criterion. Without this stop-gradient operator, training quickly collapses to a degenerate solution where the same representation is predicted for every image. This technique does not rely on sampling negative examples like in [49], and there is no need to maintain a key encoder like in [48].

1.3 Approximate Bayesian deep learning

Because DNNs are often perceived to be "black boxes", there has been particular interests in modelling the sensitivity of the model output with respect to its parameters, and Bayesian approaches are a natural fit for this task. However, computing a posterior distribution over model parameters raises particular challenges, as modern DNNs have

been scaled to hundred of billions of parameters.

Bayesian deep learning techniques using tractable variants of Monte Carlo sampling have been designed, but those methods are usually slower than gradient-based methods. Furthermore, they require an accurate implementation of a completely different optimization procedure than what is usually used in modern DNNs. This complexity and computational cost has made them a relatively niche research direction [56].

Because of this, this section focuses on approximate Bayesian deep learning. First, we briefly describe some theoretical work describing approximate Bayesian deep learning based on SGD iterates. Then, we review some articles that use a set of models to improve performance or provide uncertainty estimation, either by combining their predictions or their weights. Finally, we review a several papers that focus on estimating approximate posterior distributions over the weights of a neural network. The posterior weight distribution is assumed to be a high-dimensional Gaussian. Several ways of fitting the mean and variance of this Gaussian distribution are proposed, either based on gradient descent iterates, or estimates of the Hessian matrix of the network.

1.3.1 SGD as an Orstein-Uhlenbeck process

In a well-known paper, Mandt *et al.* [7] used tools of the stochastic differential equations (SDEs) litterature to study the stationary distribution of SGD. Let us consider a loss function over a dataset of size N that depends on model parameters θ:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} l_i(\theta),\ g(\theta) = \nabla_\theta \mathcal{L}(\theta), \tag{1.37}$$

where each l_i is the contribution to the total loss of a single sample, i.e., $l_i(\theta) = l_i(x, \theta)$. In the case of a MAP estimate, the objective is the sum of a log-likelihood and a prior, so that

$$l_i(\theta) = \log p(x_i|\theta) - \frac{1}{N}\log p(\theta). \tag{1.38}$$

For computational reasons, the full gradient $g(\theta)$ is usually not computed, but estimated on a random sample $\mathcal{J}$ of the full dataset, i.e

$$\hat{g}_S(\theta) = \nabla_\theta \mathcal{L}_\mathcal{J}(\theta), \tag{1.39}$$

where $\mathcal{L}_{\mathcal{J}}(\theta) = \frac{1}{|\mathcal{J}|}\sum_{i=1}^{|\mathcal{J}|} l_i(\theta)$. The authors [7] make a set of assumptions to view SGD as an Orstein-Uhlenbeck process:

— As the batch is a random sample of the full dataset, the stochastic gradient (and therefore the gradient noise $g(\theta) - g_{\mathcal{J}}(\theta)$) is a sum of identically distributed independent contributions. According to the central limit theorem, it follows that

$$\hat{g}_{\mathcal{J}}(\theta) \simeq g(\theta) + \frac{1}{\sqrt{|\mathcal{J}|}}\nabla g(\theta),\ \nabla g(\theta) \sim \mathcal{N}(0, C(\theta)), \tag{1.40}$$

— the covariance matrix of the gradient noise is a symmetric positive-semidefinite matrix which is approximately constant with respect to θ, i.e

$$C(\theta) \simeq C = BB^T, \tag{1.41}$$

— the difference between two successive iterates is

$$\Delta\theta(t) = -\varepsilon g(\theta(t)) + \frac{\varepsilon}{\sqrt{|\mathcal{J}|}} B\Delta W,\ W \sim \mathcal{N}(0, I). \tag{1.42}$$

Furthermore, the authors assume that this finite-difference equation can approximate the following continuous-time SDE:

$$d\theta(t) = -\varepsilon g(\theta)dt + \frac{\varepsilon}{\sqrt{|\mathcal{J}|}} BdW(t), \tag{1.43}$$

— the stationary distribution of SGD iterates is constrained to a region where the loss can be approximated in the following fashion:

$$\mathcal{L}(\theta) \simeq \frac{1}{2}\theta^T A\theta, \tag{1.44}$$

where A is positive definite matrix, the Hessian of the loss at the optimum.

Thanks to those assumption, SGD can be defined as an Orstein-Uhlenbeck process and the Gaussian stationary distribution $q(\theta)$ can be computed analytically:

$$q(\theta) \propto exp\left[-\frac{1}{2}\theta^T\Sigma^{-1}\theta\right]. \tag{1.45}$$

The authors note that those assumptions do not hold in the general case for DNNs. During early phases of training, the iterates are not located in the vicinity local mininum. It is

only where the iterates are "close" to a local minima that the assumptions used above prove correct. If we use a very long training schedule with large enough gradient steps, several local minimas might be explored during training.

1.3.2 Ensembling neural networks for performance and uncertainty estimation

In this subsection, we focus on papers [57]–[59] that use the intuitions detailed above, even if they do not explicitely estimate the parameters of the approximate posterior distribution. The authors of those papers combine several models, either by combining their predictions or their weights.

In [57], Lakshminarayanan *et al.* evaluate the performance of ensembles of neural networks for two specific goals: higher performance, and better model calibration. Calibration describes the relationship between model confidence and error rate. DNNs are usually overconfident: their predicted probability does not match their error rate. In other words, a classifier predicting a class with a probability of 0.99 does not achieve a 1% error rate on this class.

Models in the ensemble are trained in a conventional way, aside from the use of adversarial training [40]. For a set of weights θ and an image-label pair (x, y), an additional "adversarial" training example x' can be generated as

$$x' = x + \varepsilon \text{sign}\left(\nabla_x \mathcal{L}(\theta, x, y)\right), \tag{1.46}$$

Where ε is a small random value. This additional training example has the same label y as the original one but is displaced along the gradient of the loss with respect to the original input x. As the model has to predict the same label for the original training sample and the adversarial example, this technique regularizes training and makes the loss surface smoother and the classifier more robust.

In [57] The authors note that conventional models usually need additional randomness to ensure that the ensemble has enough diversity to provide an increase in performance. This randomness can be injected by training the model on random samples of the data (bagging), or by choosing a base learner with a stochastic training procedure (like random selection of features for random forest). However, the batch sampling of DNNs usually

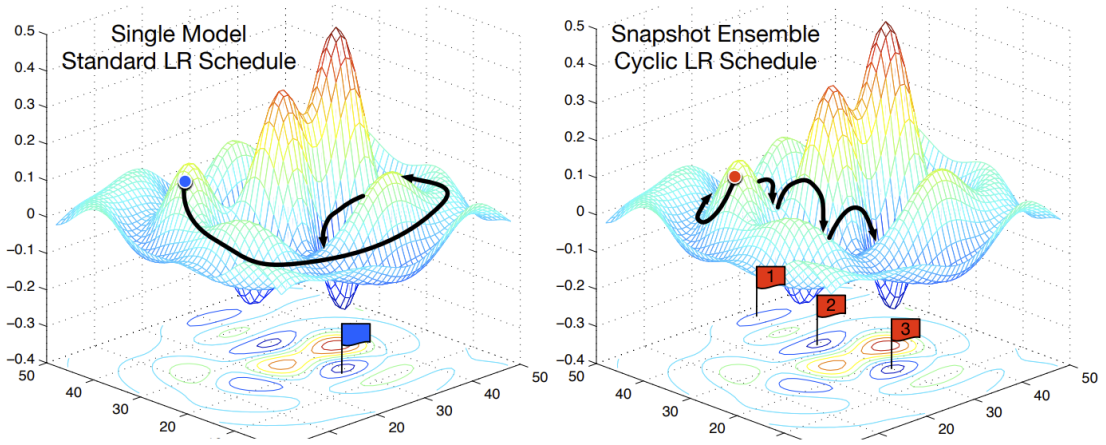


Figure 1: **Left:** Illustration of SGD optimization with a typical learning rate schedule. The model converges to a minimum at the end of training. **Right:** Illustration of Snapshot Ensembling. The model undergoes several learning rate annealing cycles, converging to and escaping from multiple local minima. We take a snapshot at each minimum for test-time ensembling.

Figure 1.13 – In [58], parameter space is explored using a cyclical learning rate schedule. An ensemble is sampled from saved checkpoints.

provides enough randomness. Models are agregated as an uniform mixture:

$$p(y|x) = \frac{1}{M} \sum_{m=1}^{M} p(y|x, \theta_m), \tag{1.47}$$

where M is the number of models in the ensemble. For classification, this aggregation method provides an improvement in performance, and improved model calibration. While the authors demonstrate the potential improvements brought by ensembling DNNs, training M different models remain costly. In [58], the authors find that a specific training schedule can be used to extract a diverse enough ensembles from checkpoint of a single training run. To ensure a thorough exploration of the parameter space, the authors use a cyclic learning rate schedule. When the learning rate is low, the model converges to a local minima, and the modelling suggested in Section 1.3.1 is correct. When the learning rate gets higher, gradient steps are larger and the model parameters escape the current local minima. Figure 1.13 illustrates this phenomenon.

While this method is computationally cheaper than training M models, the period of the cyclical learning rate remains high (around 10 epochs), so the training cost is increased. In [59], Garipov *et al.* suggest an alternative. Noticing that most local minimas are connected by curves where the loss stays low, they suggest retrieving model weights

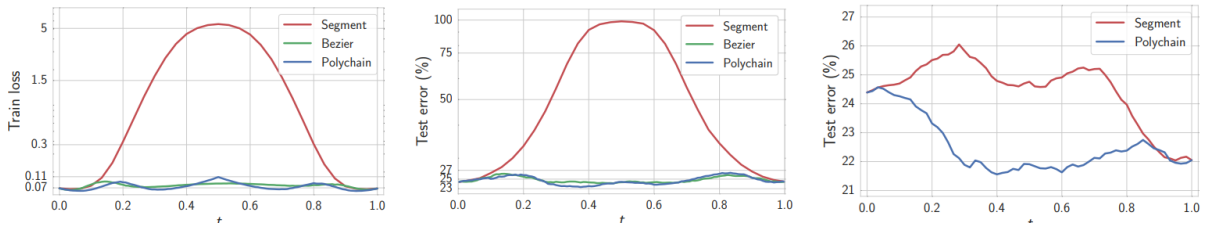


Figure 1.14 – L_2-regularized cross-entropy train loss (left) and test error (middle) as a function of the interpolation parameter t between local minimas. Straight lines between local minimas showcase a large increase in loss, while a Bezier interpolation between both local minimas shows the presence of a low-loss path between those models. "Polychain" (a polygonal chain between minimas) shows similar behavior.

along those curves. Ensembles can be built by averaging the predictions of the model retrieved along curves linking a pair of local minimas. The low loss path along this curve is shown in Figure 1.14.

The authors also confirm the insight of [58], but show that a strong ensemble can be retrieved with a cyclical learning that has a much shorter period (2 to 4 epochs), with improved performance.

All methods mentioned previously perform model agregation in prediction space. However, models can also be agregated in weight space. This is what Izmailov *et al.* demonstrate in [13]. While using an Exponential Moving Average (EMA) of SGD iterates is fairly common to stabilize training, this paper uses an uniform average of SGD iterates, called Stochastic Weight Averaging (SWA). An early example of uniform weight averaging for regularization is Polyak-Ruppert averaging [60], [61]. In this paper, the strength of this idea is demonstrated even in the context of non-convex optimization for DNNs. The main difference between SWA and Polyak-Ruppert averaging is that SWA uses a constant high learning rate instead of a decaying learning rate, to ensure parameter space exploration. The authors hypothesize that because once a local minima is reached, SGD samples the posterior weight distribution (see Section 1.3.1), the barycenter of those weights is more likely to lie near the "center" of the local minima. In this location, it is less likely that a small step in weight space will lead to a large increase in loss. In other words, this minima is "flatter", which has long been hypothesized to correlate with improved generalization [62], [63].

While all of those papers use the insights provided by Mandt *et al.* in [7] and the litterature dealing with the limit distribution of SGD in non-convex optimization in general,

none of them actualy estimate a posterior distribution over the weights of the model. We cover a few references dealing with this problem in the next section.

Approximating a posterior distribution with SGD iterates

In [6], Maddox *et al.* approximate the posterior distribution over the model weights θ as a Gaussian distribution, and call this method Stochastic Weight Average Gaussian (SWAG). The mean of this distribution is estimated with the running mean of SGD iterates, i.e the SWA solution θ_{SWA} as described in [13]:

$$\theta_{SWA} = \frac{1}{T}\sum_{i=1}^{T}\theta_i. \tag{1.48}$$

Several covariance approximations are possible, although computational costs are an issue, as the size of the covariance matrix scales quadratically with the number of parameters. Because of this, a diagonal covariance can be used:

$$\Sigma_{\text{diag}} = \text{diag}(\bar{\theta^2} - \theta_{SWA}^2), \tag{1.49}$$

with $\bar{\theta^2} = \frac{1}{T}\sum_{i=1}^{T}\theta_i^2$. This diagonal covariance model can be enriched with an additional low-rank covariance matrix. This approximation has the following form:

$$\Sigma_{lr} = \frac{1}{K-1}\sum_{i=1}^{K}(\theta_i - \theta_{SWA})(\theta_i - \theta_{SWA})^T = \frac{1}{K-1}DD^T. \tag{1.50}$$

Note that as θ_{SWA} is not available until the end of training, it is approximated during training by $\bar{\theta_i}$, which is the running mean of the first i SGD iterates. D is a deviation matrix with K columns of value $\theta_i - \bar{\theta_i}$. The number of columns K of D (the number of considered SGD steps) is an hyperparameter. Both covariance approximations can be combined, so that the final posterior approximation is $\mathcal{N}(\theta_{SWA}, \frac{1}{2}(\Sigma_{\text{diag}} + \Sigma_{lr}))$. One can sample from this approximation with the following equation:

$$\tilde{\theta} = \theta_{SWA} + \frac{1}{\sqrt{2}}\Sigma_{\text{diag}}^{1/2}z_1 + \frac{1}{\sqrt{2(K-1)}}Dz_2, \tag{1.51}$$

where $z_1 \sim \mathcal{N}(0, I_d)$ and $z_2 \sim \mathcal{N}(0, I_K)$ where d is the number of parameters in the network. This approximate posterior can be used to approximate a Bayesian Model Average (BMA) [64]:

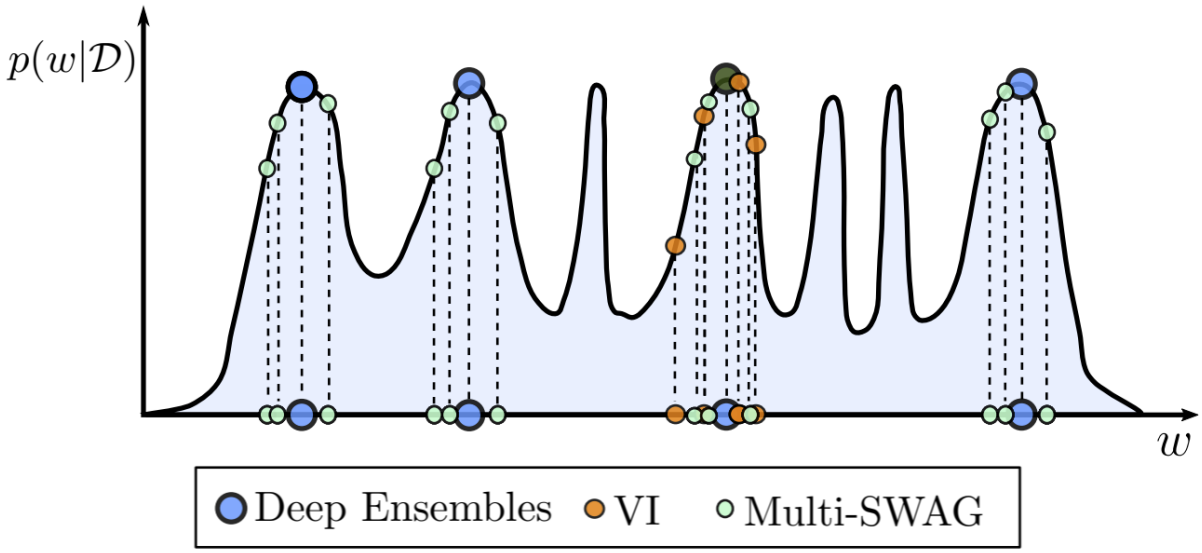


Figure 1.15 – Comparison of the sampling of the posterior distribution for several methods. An ensemble of several DNNs trained separately [57] effectively provides several true samples of the posterior distribution. However, the number of samples is low. SWAG can cover a local minima of the loss function (a mode of the posterior distribution) much more densely, but can only cover one mode. Multi-SWAG covers several modes of the posterior distribution by fitting several Gaussian distributions during training, to form a mixture model of the posterior distribution.

$$p(y_t|\mathcal{D}, x_t) = \int p(y_t|\theta, x_t)p(\theta|\mathcal{D})d\theta, \tag{1.52}$$

where $\mathcal{D}$ is the data distribution, (x_t, y_t) is a test input-output pair and $p(\theta|\mathcal{D})$ is the posterior weight distribution. This integral is approximated with samples of the approximate posterior distribution estimated above:

$$p(y_t|\mathcal{D}, x_t) \simeq \frac{1}{S}\sum_{s=1}^{S} p(y_t|\theta_s, x_t),\ \theta_s \sim p(\theta|\mathcal{D}). \tag{1.53}$$

Ensembles generated from the posterior weight distribution bring improvements in terms of perfomance and model calibration, and do so at a lower compute cost than ensembles composed of separately trained models. However, true posteriors of neural networks are very unlikely to be unimodal. The stationary distribution proposed in Mandt *et al.* [7] is only relevant in the vicinity of a local minima. Therefore, while the approximation proposed above provides performance improvements, it does not model this multimodality. In Maddox *et al.* [6], the authors improve SWAG by representing the posterior distribution

as a mixture of Gaussians, centered in several different basins of attraction of the loss surface. Figure 1.15 provides a schematic summary of the idea.

1.4 Object detection with deep learning models

Object detection is most commonly defined as fitting axis-aligned bounding boxes around objects of interest. This task is relevant in a number of fields, like autonomous driving or medical imaging. Bounding boxes have an associated class label, so that any object of interest is localized and classified. In two-stage object detectors, the detection task is divided into three subtasks. First, there is a region proposal step, where a model predicts bounding boxes around foreground objects. Then, a bounding box regression model refines those proposals, using the content of the region proposal. Finally, a classification networks predicts a class label for the bounding box.

One-stage detectors do not have any region-proposal. This is done to improve inference time, but usually lowers detection performance. In this section, we will review two object detection models and explain how they solved the most common problems associated with this task. First, we look at Faster R-CNN as a well-known example of a two-stage object detector. Then, we study CenterNet as an example of a one-stage object detector, as it is very simple to implement which makes it especially interesting.

1.4.1 Object detection with Region-CNN

Fast R-CNN

In "Fast R-CNN", Girshick *et al.* [65] describe the basic components of their object detection model. Figure 1.16 gives a summary of the architecture. Fast R-CNN region proposal step is separated from the object detection model. At the time, Selective Search was used for region proposal [66]. Therefore, the complete architecture could not be trained end-to-end.

For a given image, a feature map is produced by a ConvNet which is usually called a backbone. Given that region proposals might have different size, the subsets of the feature map corresponding to those region proposals will also have different sizes. To ensure that features corresponding to those region proposals all have the same size, a Region of Interest (ROI) pooling layer is used. This adaptive pooling layer agregates a feature map to a constant output size, no matter the size of the input. Once this is

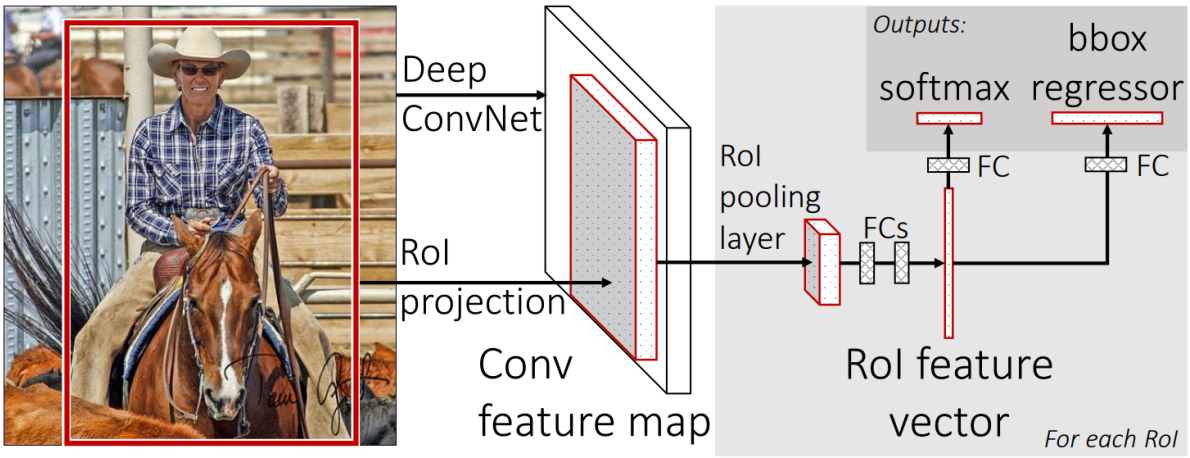


Figure 1.16 – For a proposed region, features from a ConvNet are retrieved. An adaptative pooling mechanism called ROI pooling ensures that features have the same spatial size, no matter the size of the proposed region. Pooled features are used for classification and bounding box regression. (*Source*: [65])

done, two small neural networks are used for bounding box regression and bounding box classification. The object detector is trained with the following criterion:

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda 1_{u \geq 1} L_{loc}(t^u, v) \tag{1.54}$$

with p the predicted probability distribution over the classes, and u the one-hot vector describing the true class label. L_{cls} is a cross entropy loss. $t^u = (t^u_x, t^u_y, t^u_h, t^h_w)$ is the predicted bounding box regression vector and v the true bounding box vector. The first two coordinates of the bounding box vector are the coordinates of the top right corner. The last two coordinates indicate the height and width of the bounding box. The bounding box regression loss is defined as the following robust loss:

$$L_{loc}(t^u, v) = \sum_{i \in \{x,y,h,w\}} \text{smooth}_{L1}(t^u_i - v_i) \tag{1.55}$$

with smooth_{L1} defined as:

$$\text{smooth}_{L1}(r) = \begin{cases} \frac{1}{2} r^2 & |r| < 1 \\ |r| - \frac{1}{2} & |r| > 1. \end{cases} \tag{1.56}$$

Note that this bounding box regression loss is only active if the considered region does

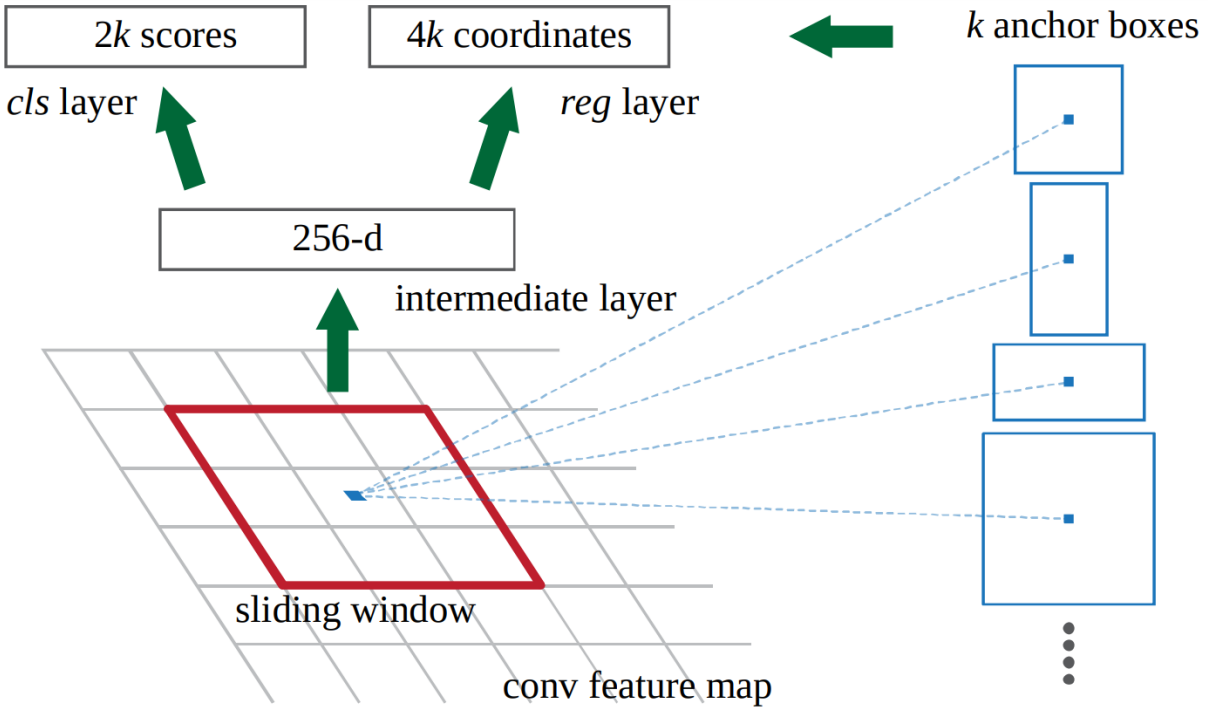


Figure 1.17 – For every pixel of the feature map produced by the backbone, the RPN produces a set of regression vector modifying the pre-defined anchors, and a set of objectness scores which quantify how likely each anchor is to contain an object. (*Source*: [9])

not belong to the background. During training, background examples (regions with a small intersection with a ground truth bounding box) are sampled to ensure that the classification network accurately classifies the background.

Faster R-CNN

In [9] Ren *et al.* suggested training a model for regression proposal, called a Region Proposal Network (RPN) so that the entire object detection model could be trained end-to-end. This network uses the same backbone as the object detection model (see Section 1.4.1 for an explanation). Figure 1.18 shows the architecture of Faster R-CNN. For every pixel of the feature map, there are k (where k is a tunable hyperparameter) anchor boxes, usually with different aspect ratios or size, as shown in Figure 1.17. This is mostly because the resolution of the output feature map is relatively low, so that a single anchor per pixel would be insufficient to accurately locate every object in the image. Those anchors act as pre-defined region proposals. The RPN refines those anchors by predicting a regression vector (like the object detection model). Furthermore, the RPN predicts an objectness

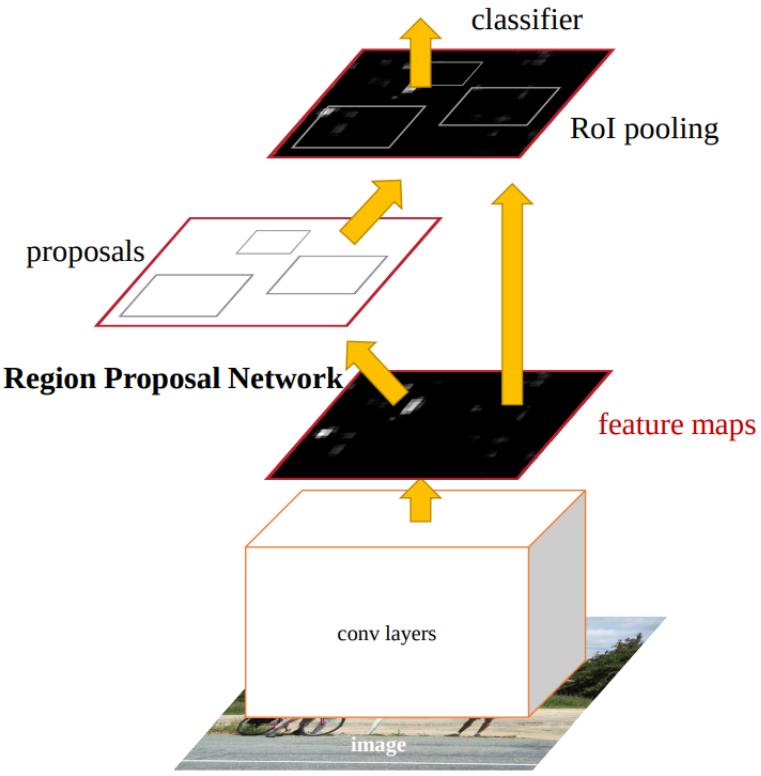


Figure 1.18 – Architecture of Faster R-CNN. A ConvNet produces a feature map for an input image. This feature map is used to produce region proposals. Those region proposals are selected, and an object detection network is trained as presented in Figure 1.16. (*Source*: [9])

score for every anchor. This score corresponds to the likelihood that the current bounding box contains an object, instead of background.

The RPN is trained like the object detection model, by minimizing the sum of a classification and regression loss. Here, the classification task is the rejection of background. While this idea makes Faster R-CNN considerably faster than Fast R-CNN, there an issue. For 2 adjacent pixels of the feature map at the output of the backbone, 2 anchors can be adjusted to cover the exact same subset of the image. In other words, several regions with a high objectness score can be proposed for a single object.

To deal with this issue, the set of region proposals is reduced by thresholding the objectness score. Furthermore, overlapping boxes in the set of object proposals are merged using Non-Maximum Supression (NMS). If two boxes have high overlap (defined by their Intersection Over Union, or IoU), only the one with the highest objectness score is kept. This method is effective, but requires setting another hyperparameter, and the enumer-

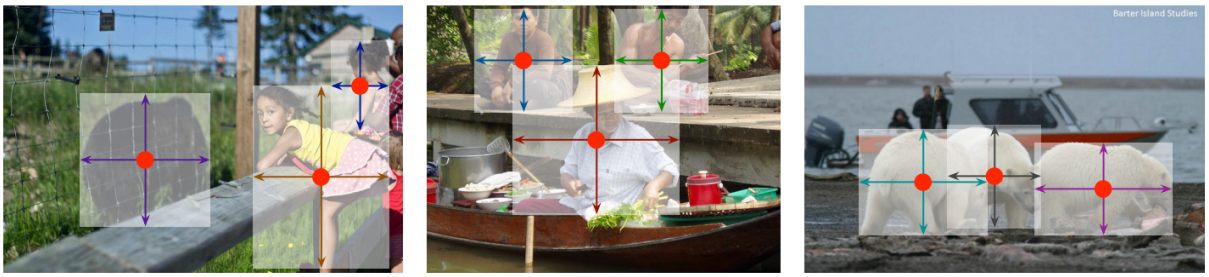


Figure 1.19 – Output of the CenterNet model. The center of the bounding box is predicted, a specific heatmap is predicted for every class in the dataset. For every location of the heatmap, a (height, width) tuple is predicted, representing the size of the bounding box at this location. (*Source*: [67])

ation of region proposals in two-stage detectors is computationally costly. However, this class of object detection models usually reaches a high level of performance. In the next section, we discuss a one-stage object detector that does not require region proposal or NMS.

1.4.2 NMS-free object detection: Object as Points

In [67] Zhou *et al.* proposed a simple one-stage object detector that does not need NMS nor any pre-defined anchors. Instead, the object detection model (named CenterNet) is implemented as a ConvNet with three output branches. For an image of size $H \times W$, a first branch outputs a heatmap of size $H/K \times W/K \times C$ (with K a downsampling factor, and C the number of classes) predicting the center of bounding boxes. A second branch predicts the size of the bounding boxes as a (height, width) tuple for every location u, v in the output of the heatmap branch, with $u, v \in \{1, \cdots, H/K\} \times \{1, \cdots, W/K\}$. Figure 1.19 gives a visual depiction of the output of CenterNet. Bounding box height and width are predicted for every location of the image, even when this location does not contain any object. Bounding box height and width are only retrieved where the output of the bounding box center prediction branch exceeds a pre-defined threshold.

Finally, the heatmap output has resolution $H/K \times W/K$, which is lower than the resolution of the input image. Therefore, a third output branch is used to predict an offset for every location $u, v \in \{1, \cdots, H/K\} \times \{1, \cdots, W/K\}$ so that the model can predict bounding box centers with sub-pixel accuracy.

Training

The heatmap branch is trained by minimizing a modified cross entropy loss called Focal Loss [68], [69] between its output and a heatmap of Gaussian "spots" laid over the center of bounding boxes. While one could predict a binary image where bounding box centers would take value 1 and the background value 0, bounding box centers would be too under-represented. Therefore, Zhou *et al.* [67] choose to blur the heatmap with a Gaussian kernel.

Local maximas of the heatmap prediction are retrieved. Those location corresponds to the predictions of the model. Bounding box size and offset are retrieved in the outputs of the two other branches. Bounding box and offset errors are computed only in those locations. Both branches are trained by minimizing a L1 distance between prediction and ground truth.

1.5 Conclusion

Modern deep learning has received considerable attention in image analysis. Extremely flexible architecture can be trained end-to-end with simple criterions, using backpropagation and stochastic gradient descent. In spite of the non-convexity of commonly used training criterions because of the non-linearities in this class of models, finding "good" local minimas is relatively easy because of implicit regularization.

Bringing the benefits of this class of high-performing architectures to biomedical imaging is more difficult, as labelled data is rare. SSL lets us pre-train models without any labels, either by learning to generate images or by learning invariance to common image augmentation techniques (noise, cropping, flipping, etc).

Because of the black box nature of modern DNNs, Bayesian deep learning has been an active field of research. In approximate Bayesian deep learning, the posterior distribution over the weights of the model is usually modelled as Gaussian. The parameters of this distribution can be estimated with approximate samples, like SGD iterates.

Finally, DNNs can be used to build object detection models. Those models are usually divided in two categories: one stage and two stage detectors. Two stage detectors are more complex, but reach a higher level of performance. In Chapter 4, we explore the potential of one-stage object detection models for chromosomal aberration detection.

Chapter 2

Image analysis for the automation of biological dosimetry

In the first part of this chapter we introduce the basics cytogenetic biological dosimetry. We review some biological factors that are especially relevant from a computer vision perspective as they cause variations in image quality. Next, we review historical examples of chromosomal aberration detection methods. We focus on a commercial solution developed by canadian researchers as its inner workings were explained in details in a series of publications.

Finally, we review some references written by researchers at LRAcc dealing with evaluating another commercial solution in a semi-automated regime. While the computer vision techniques used in this solution were not described in the open scientific litterature its evaluation by biological dosimetry specialists provides a benchmark to compare against for the new methods developed in this thesis.

2.1 Basics of biological dosimetry

The goal of biological dosimetry is estimating a dose of ionizing radiation using biomarkers instead of measuring it with instruments like a dosimeter [14]. This may be necessary because dosimeter readings are unavailable, in the case of an accidental (or potentially malicious) exposition. While several biomarkers of radiation have been identified, the average number of DC per peripheral blood lymphocytes (also called dicentric yield) is the gold standard according to the IAEA [14].

Those aberrations can be observed during the metaphase step of mitosis, so that an image of a peripheral blood lymphocyte undergoing metaphase is usually abbreviated to "metaphase". Once a blood sample is collected, blood cells are grown and Demelcocine is used to stop mitosis during metaphase. The blood sample is spread on a transparent slide and observed with a bright-field microscope. The slide is backlit, and because chromo-

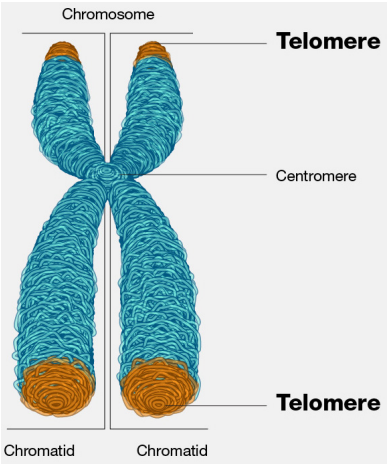
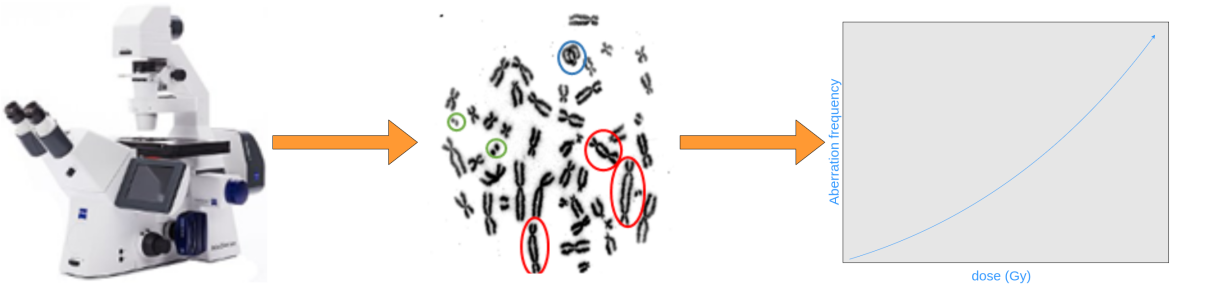


Figure 2.1 – Basic morphology of a metaphase chromosome.

somes are translucent objects, Giemsa staining is used to make them opaque and improve contrast with the bright background [70], [71] (see Figure 2.2).

DCs have two centromeres, which is the visible constriction of its chromatides (like at the center of a MC). For an example of a DC, see the shape circled in red in Figure 2.2 b). The presence of DCs and fragments in a metaphase (circled in green in Figure 2.2 b)) are highly correlated. The required cell culture duration for DC scoring is lower than for other methods like micronuclei scoring [70]. Furthermore, the preparation process is simple and relatively cheap. Other imaging modalities, like Fluorescence In Situ Hybridization (FISH) offer significant advantages: probes can be used to highlight other types of aberrations like translocations, or centromeres [72], [73]. However, FISH is significantly more time consuming, and more expensive. MCs and DCs can be discriminated using their morphological differences.MCs are X-shaped (see Figure 2.1), while DCs look like a pair of MCs bound together.

The number of DCs in a metaphase is random. It follows a Poisson distribution, where the parameter λ of this distribution is the average number of DC per cell (also called dicentric yield) for the considered dose. This Poisson model is adequate in the case of a whole body photon exposition. The relationship between DC yield and dose (in Grays) is described by a calibration curve with a linear-quadratic shape. In the case of a partial exposition, only some blood cells are exposed to ionizing radiation. The blood sample is usually modelled as a mixture of an irradiated and a non-irradiated sample [74]. This effectively increases the number of cells without any DC, and partial expositions are often modelled with Zero Inflated Poisson models. In this context, the DC distribution



a) metaphase acquisition b) aberration counting c) dose estimation with calibration curve

Figure 2.2 – Chronology of a biological dosimetry-based estimation of an ionizing radiation dose. First, a blood sample is taken and cells are grown for 48 hours, Demelcocine is used to stop cells in the metaphase stage of mitosis. Metaphase acquisition is automated with modern microscopy systems, as shown in a), and chromosomal aberrations are counted, as displayed in b). The proportion of chromosomal aberration is linked to the ionizing radiation dose through a linear-quadratic relationship, see c).

showcases overdispersion: the variance of the distribution is greater than the mean.

For a blood sample that was not exposed to ionizing radiation, the dicentric yield is around 10^{-3}: the vast majority of cell does not contain any aberration. For small doses, a large number of metaphases has to be processed to accurately detect a deviation from the basal aberration rate. This leads to stringent requirements in terms of false positives, to prevent overestimation of low doses. Even at 5Gy, which corresponds to the median lethal dose for a whole body exposure (also called LD_{50}), there is only around 1 aberration per cell. This means that in a single metaphase, on average, 45 chromosomes are healthy and a single DC is observed.

To estimate dicentric yield, a sample of metaphase is checked for chromosomal aberrations, a process called DC scoring. In practice, human experts perform binary classification on each chromosome, sorting them between the MC and DC class. Furthermore, multicentrics chromosomes and fragments are identified. To this end, those experts undergo several months of training and are then evaluated on benchmark datasets. This training helps humans achieve an extremely low error rate. However, this error rate comes at a cost: human can process a relatively low number of metaphases per hour (see Figure 2.3). Cognitive load is high, and human experts can only count chromosomal aberrations for a few hours at a time to keep the error rate low. Furthermore, a large number of metaphase is needed to achieve usable confidence intervals for low doses, as illustrated in Figure 2.3.

Within the context of a large scale exposure there may not be enough human experts to accurately triage patients between different therapeutical options. In this case, com-

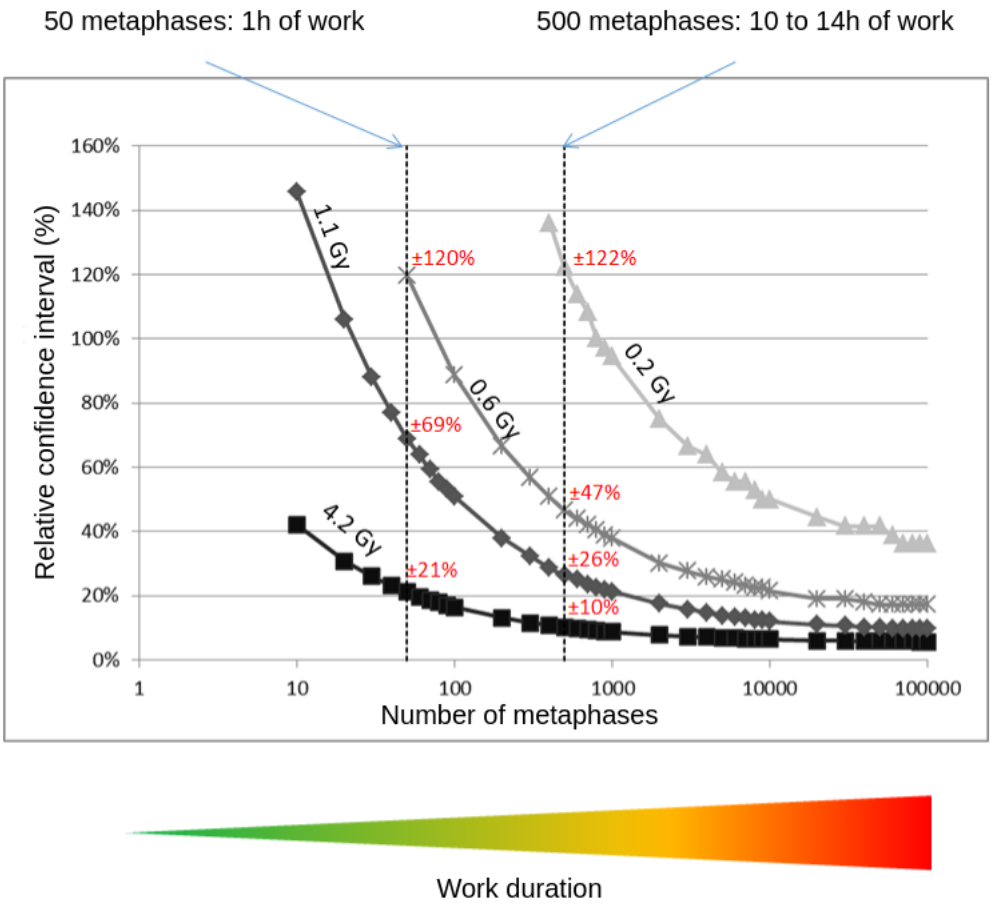


Figure 2.3 – Relative confidence interval size as a function of metaphase number and dose. The lower the dose, the higher number of images is needed to get a narrow confidence interval. Aberration counting takes about one hour for 50 images, and a thousand image may be needed for a narrow enough confidence interval for a small dose.

puterized expertise could provide invaluable help. This is the central motivatoin for the deep-learning-based ADS proof of concept presented in this thesis.

2.2 Challenges of automated aberration counting

In Figure 2.2 a), the DC circled in red is relatively obvious even to the untrained eye. Unfortunately, chromosomes are flexible objects and the chromatids of a MC may fold in way that would make a MC indistinguisable from a DC. Human experts can shift the focal plane of the microscope to disambiguate those cases, as folded chromatids and

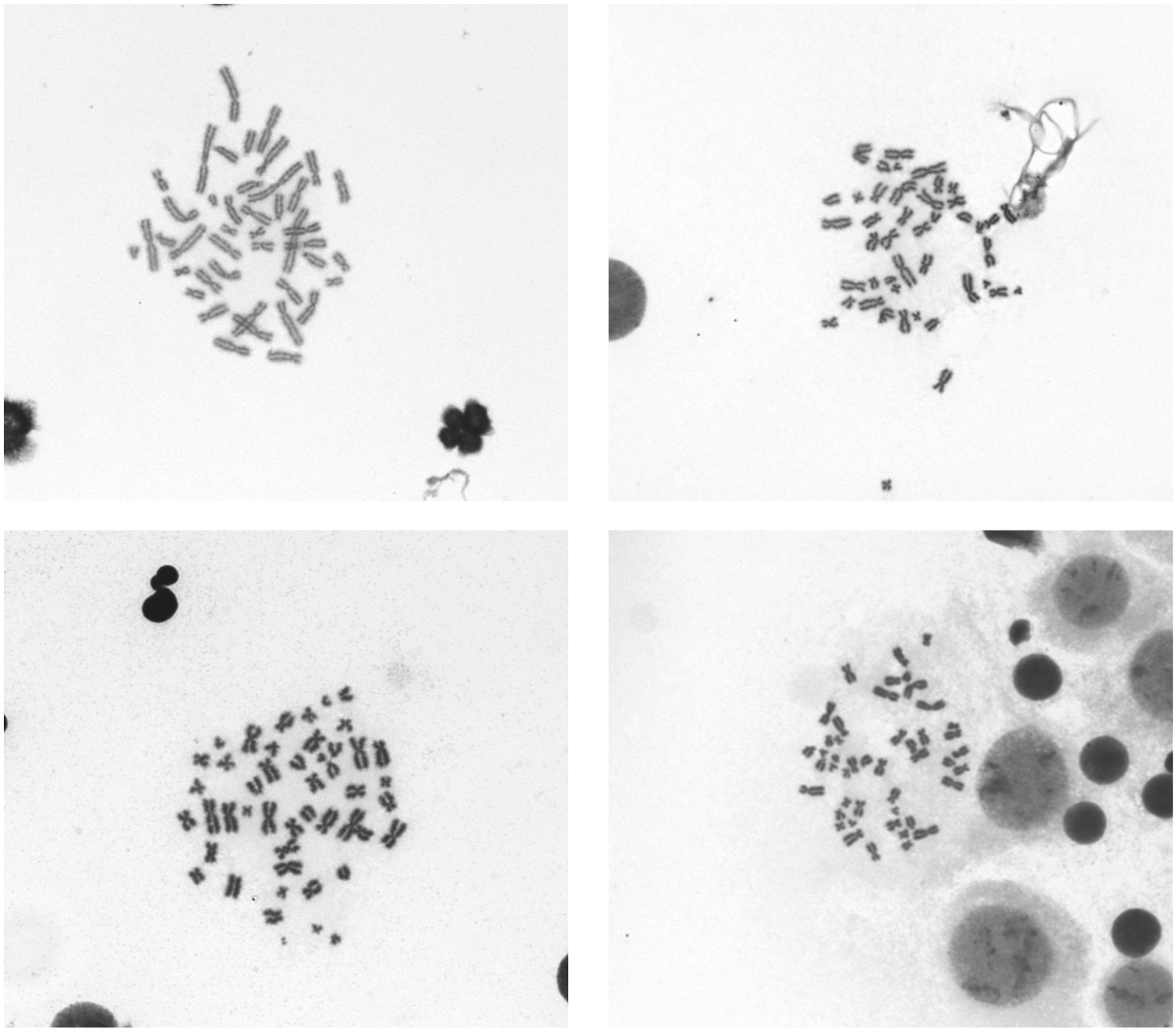


Figure 2.4 – Four examples of metaphases showcasing variations in acquisision conditions. Variations in chromosome spread, chromosome condensation, chromosome thickness, chromosome texture and illumination are visible. Furthermore, some metaphases contain more debris (cellular nuclei, filaments) than other.

centromeres have different thickness. This can then be used to distinguish MCs from DCs, as DCs have two centromeres. Because the microscope-mounted camera only acquires 2D data, this information is not available in the data we are working with.

Beyond those issues, the process of cell culture and metaphase acquisition leads to variation in image quality. In each cell, mitosis is stopped at a different level of chromosome condensation, which leads to difference in chromosome size or texture. Giemsa staining may bind to the chromosomes in a non-uniform manner. The location of the cell on the

glass slide leads to variations in backlight intensities between metaphases, or even in a single metaphase, so that a gradient in backlight intensity could be observed even for a single cell. A lot of metaphases contain cellular debris, which might be circular and easy to reject but could also be filament-shaped and much harder to discriminate from a chromosome fragment. In Figure 2.4 a few examples of metaphases showcasing those variability factors in the image acquisition process are shown.

Any computer vision system aiming to accurately count chromosomal aberrations in those conditions will need to deal with those sources of variability, either in an explicit or implicit manner.

2.3 An historic example of automated dicentric scoring systems

While biological dosimetry is a relatively niche field of study, it quickly became of playground for advances in computer vision and image processing. One of the earliest implementation of an ADS was called the Edinburgh dicentric hunter. It was introduced in "Radiation dosimetry by automatic image analysis of dicentric chromosomes" by Bayley *et al.* [70] in 1991. Because of the lack of sufficiently effective computer vision tools, the authors choose to significantly restrict the range of DC they would try to detect. For example, they do not attempt to detect DCs where the centromeres are very close to each other, or DCs where one of the centromeres is located at the edge of the chromosome (acrocentric chromosomes). Based on previous references, the authors assume that those configurations amount to around 30% of all DCs.

The authors also note that one of the significant challenges in evaluating an ADS is building a ground truth. Indeed, there is usually a high level of inter-observer disagreement during a manual scoring session of DCs. In [70], DC scoring is divided in 5 steps:

— metaphase finding;
— image acquisition;
— segmentation of chromosomes;
— MC - DC classification;
— human review.

The metaphase finding step was implemented with the Cytoscan 110 metaphase finder, which was evaluated in [76]. During metaphase acquisition, the images undergo global thresholding. The software attempts to identify chromosome objects, reject debris and

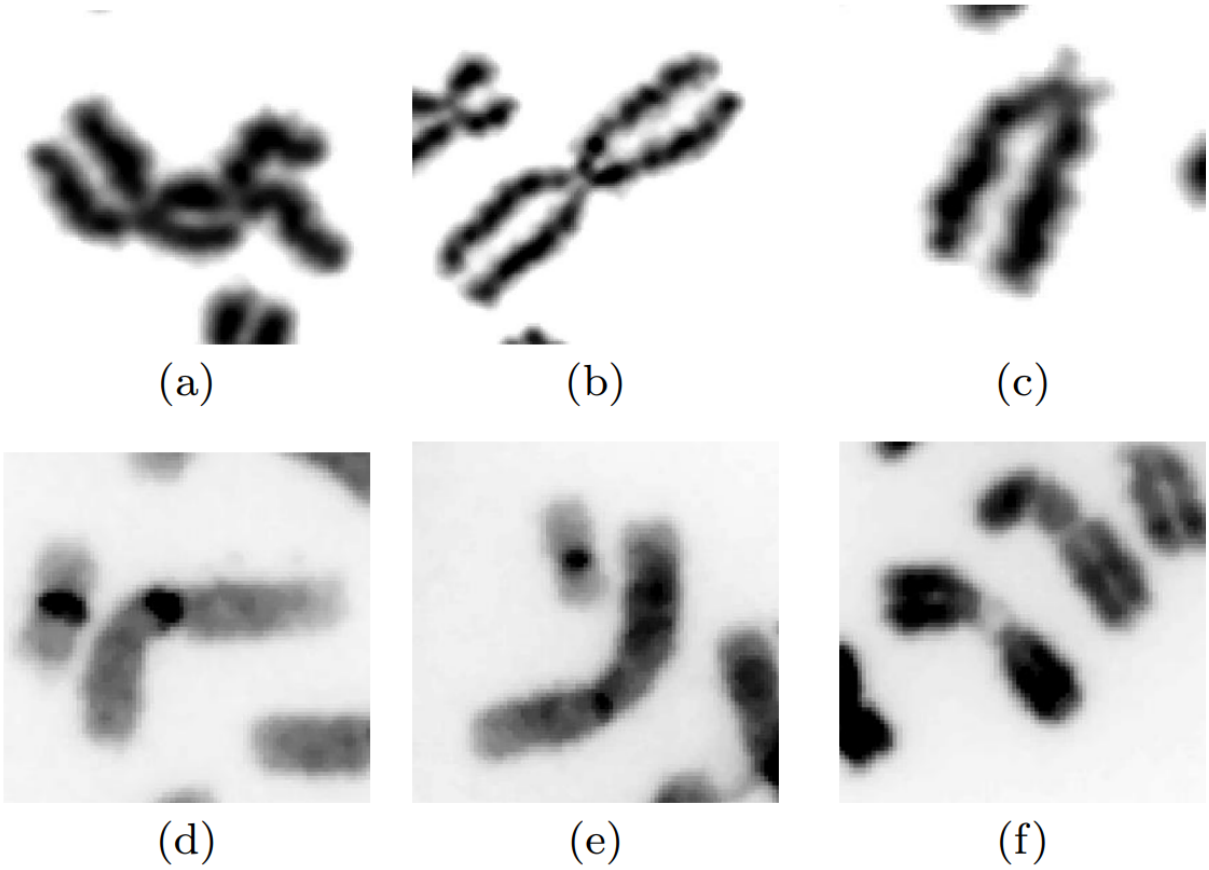


Figure 2.5 – This figure depicts various degrees of SCS present in some Giemsa stained chromosome cell images (a), b) anc c)) as well as some lenghty chromosomes characteristic to those prepared at a cytogenetic laboratory (d), e) and f)). (*Source*: [75])

split chromome clusters. Metaphases are accepted or rejected on the basis of their object counts. Early and late metaphase cells are rejected by identifying the level of chromosome condensation based on their mean aspect ratio. This prevents the selection of metaphases containing chromosomes with excessive Sister Chromatide Separation (SCS). Sister chromatides are the "branches" of a chromosome. Depending on the preparation of the blood sample, various levels of SCS might be encountered, as shown in Figure 2.5.

Once metaphase finding is completed, chromosome classification is performed. The classifier used by [70] is described in [77]. A first-stage classifier extracts a set of centromere candidates by retrieving points of contour constriction along the chromosome centerline. Because of this, SCS can lead to spurious DC detections as constrictions points might not correspond to centromeres. As the authors point out, chromosome shape variability is usually much higher between cells than inside a single cell. Therefore, the centromere clas-

sifier must be adapted to every cell. This can be achieved by finding a within-chromosome or within-cell normalization for feature values. A feature set is extracted at candidate points, as illustrated by Figure 2.6.

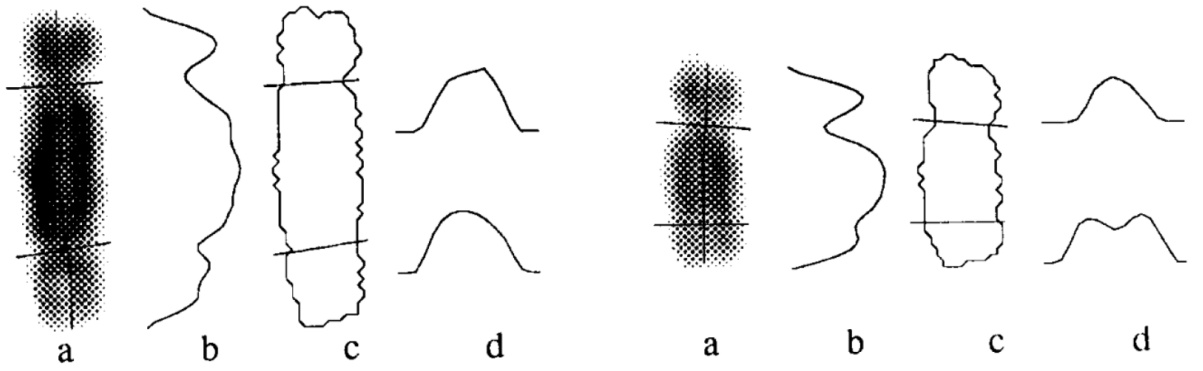


Figure 2.6 – Chromosomes showing the various structures from which feature values are computed: (a) the chromosome and its computed symmetry axis, with candidate centromere positions marked by horizontal lines, (b) the longitudinal profile of densities, (c) the chromosome boundary from which curvature values are obtained, (d) the "crossing profiles", the density profiles across the chromosome at the centromere candidates. The right-hand chromosome has a false candidate, clearly distinguished by the bimodal crossing profile. (*Source*: [70])

The set of centromere candidates is re-classified by a second-stage classifier which takes this feature set as an input. The set of detected DCs is stored, and presented to a human expert for review. The software retains the position of DCs in every cell, which makes human review with a microscope (instead of on an image acquired by a camera) possible. In practice, this possibility was found to increase analysis time, as human experts tended to prefer lengthy reviews using the microscope over fast review using DCs candidates extracted by the software.

In terms of performance, The Edinburgh dicentric hunter achieves a false positive rate of around 0.5 DC per cell according to Bayley *et al.* The system has a recall of around 40%. The sensitivity for low doses is insufficient, which is not surprising given that no attempt is made to detect around 30% of DCs. The results of Bayley *et al.* [70] were confirmed by another evaluation of the same software in Finnon *et al.* [78].

To summarize, ADS for Giemsa stained images data back to the mid 1980s. The Edinburgh dicentric hunter is an noteworthy example of early ADS dating back to 1991. The overall workflow [70] by Bayley *et al.* is still used by the modern method presented in Section 2.4. Other ADS systems have been designed with FISH imaging, which en-

ables labeling centromeres or specific chromosomes. For example, Roy *et al.* [71] made a comparison of ADS which includes fluorescence-based systems and Huber *et al.* [73] did technical review of a version of Metafer for fluorescence imagery.

2.4 ADCI: implementing an ADS framework

This section aims to provide an overview of a comercially available ADS solution developped by Cytognomix called Automated Dicentric Chromosome Identifier (ADCI). The first version of this dose estimation pipeline was described in "Towards large scale automated interpretation of cytogenetic biodosimetry data" by Li *et al.* [79] in 2012. While there are other comercially available ADS, we have more insight in the inner workings of ADCI through a series of paper providing details of its implementation.

ADCI does not use deep-learning-based object detection. DC counting is framed as two successive tasks: chromosome detection, and chromosome classification. A large amount of prior knowledge is used to solve those tasks. Metaphases are selected based on the number of objects, so that images with overlapping objects are rejected, chromosome contours are identified accurately for precise centerline extraction, centromeres are detected for accurate DC classification, etc.

2.4.1 Chromosome feature extraction

First, an accurate chromosome extraction algorithm is needed. Noise may lead to incorrect chromosome extraction when global thresholding algorithms (like Otsu thresholding [80]) are used, like in [70]. Accurate segmentation should rely on more than a single pixel for its decision. In [81], the authors found that an active contour method called Gradient Vector Flow (GVF) provided good segmentation results. Active contours ensure that the chromosome segmentation method has a relatively regular boundary.

Once an accurate chromosome contour (and segmentation) has been extracted, the centerline of the chromosome must be estimated. Usually, skeletonization is performed using morphological operators (e.g. thinning).

This step emphasizes the importance of an accurate contour extraction, as the estimated centerline location depends on the width of the estimated contour at every location of the chromosome boundary. Furthermore, biological processes can make centerline extraction difficult, the most likely being Sister Chromatide Separation (SCS) in late

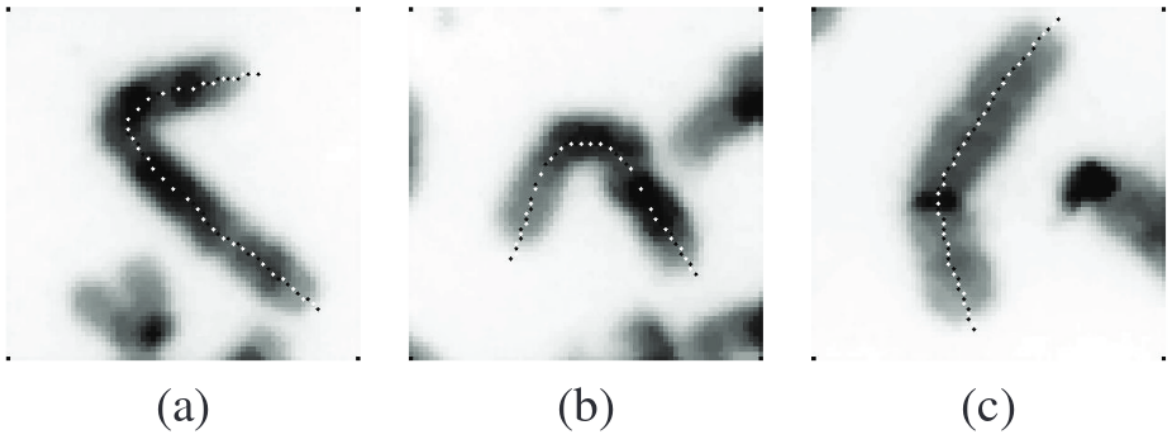


Figure 2.7 – Chromosome centerlines proposed by the GVF + DCE algorithm proposed in [81]

metaphase images. A further refinement of centerline extraction was introduced in [82].

Spurious branches may be produced during centerline extraction, especially if the chromosome is bent. In those cases, the authors performed skeleton pruning with a Discrete Curve Evolution (DCE) algorithm which was proposed in [83]. As explained in [81], this centerline extraction algorithm relies on the adjustment of hyperparameters for the snake algorithm. Those were set empirically. Figure 2.7 shows a few examples of extracted centerlines.

This centerline can be used to estimate centromere location and number, which is a common technique in biological dosimetry (see Section 2.3). This is done by building a width profile of the chromosome around the centerline by retrieving the length of scan lines that are perpendicular to the centerline.Indentifying constrictions in this width profile provides a way to count centromeres, as shown in Figure 2.8.

2.4.2 Chromosome classification

In the first ADCI prototype introduced by Li *et al.* in 2012 [79], monocentric-dicentric (MC-DC) classification relied on counting the minimas of the chromosome width profile, like in early ADS systems like the Edinburgh dicentric hunter introduced by Bayley *et al.* in 1991 [70].

The MC-DC classification was improved in Subasinghe *et al.* and Li *et al.* in 2016 [75], [85]. For centromere detection, a contour is extracted using GVF and DCE, and the

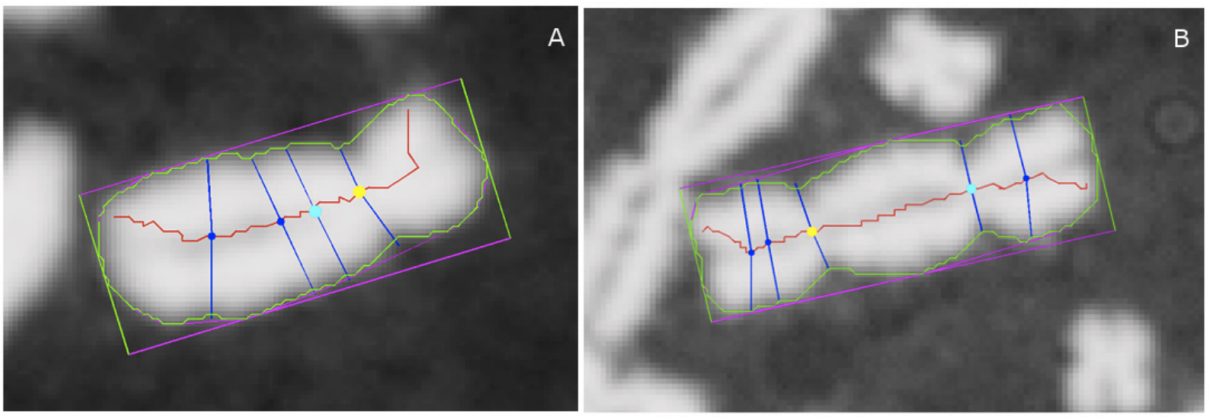


Figure 2.8 – Chromosome images processed by ADCI, annotated with key segmentation features. (A) MC and (B) DC. Chromosome contour overlaid in green, long-axis centreline in red. For reference, the minimum bounding box of the contour is also displayed in magenta and green. Yellow and cyan markers on the centerline indicate the top-ranked and 2nd-ranked centromere candidates, respectively, and all other candidates are indicated with a dark blue marker. For each centromere candidate, their corresponding width traceline (crossing through the candidate and running approximately orthogonal to the centerline) are displayed in dark blue. The arc lengths of width tracelines running down the centerline (not all shown) are used to construct a chromosomal width profile. Note that for the MC (A), the top-ranked candidate correctly labels the true centromere location, while the 2nd-ranked candidate labels a minor non-centromeric constriction. Meanwhile, for the DC example (B), both the top and 2nd-ranked candidates label true centromere locations. By comparing features extracted from the top 2 candidates (including width and pixel intensity information), the software determines if the chromosome is a MC or a DC. (*Source*: [84])

centerline is extracted, as introduced in [82]. Local minimas of the width profile of the chromosome along this centerline are considered to be candidate points for centromeres. A set of handcrafted features is build from those locations, like local curvature of the contour, thickness of the chromosome at this point, distance from the candidate to the closest end of the centerline, etc. Those features are used by a SVM classifier to accept or reject candidate points as centromeres. The distance to the separating hyperplane is used as confidence measure, called Candidate Based Centromere Confidence (CBCC). For every chromosome two candidate points are extracted, as shown in Figure 2.8. Additional steps were taken by Liu *et al.* [84] to reduce the number of false positive in MC-DC classification. A set of filter was designed to correct SVM classifications. For example, if the surface area of a candidate DC is too low, it should be rejected. Filters based on chromosome maximum and median width, surface area, oblongness were designed.

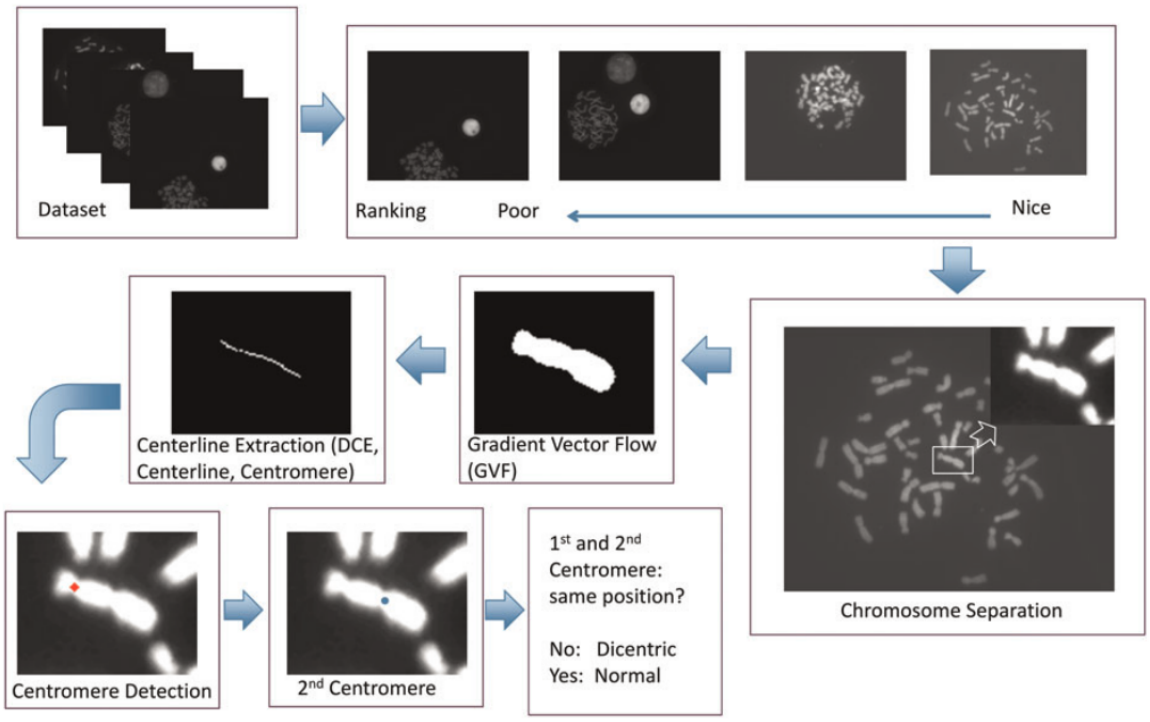


Figure 2.9 – Outputs of the 2014 version of ADCI at different steps of the pipeline as shown in [86]

2.4.3 A pipeline for dose estimation

Several iterations of the ADCI software have been published. All of them follow roughly the same outline, but ADCI was improved over time to deal with its initial shortcomings. Using the centerline extraction algorithm and the centromere detection method proposed by Subasinghe *et al.* in 2010 [81], [87], Li *et al.* [79] introduced the first prototype of ADCI and its parallelization over a compute cluster to deal with mass exposition in 2012. Particular care was taken to ensure that the results are as independent as possible of preparative methods in biodosimetry labs, as this is identified as a core factor of ADS performance variation. Using the centerline extraction improvements proposed in [82], the first iteration of the ADCI software beyond the prototype stage was introduced by Rogan *et al.* in 2014 [86]. A flowchart of this pipeline is visible in Figure 2.9.

A new version of ADCI was introduced by Rogan *et al.* [88] in 2016. This version uses an improved centromere location algorithm and MC-DC classifier introduced by Subasinghe *et al.* and Li *et al.* in 2016 [75], [85]. This version of ADCI is compared to human evaluation in terms of calibration curve fitting. First, the authors note that the

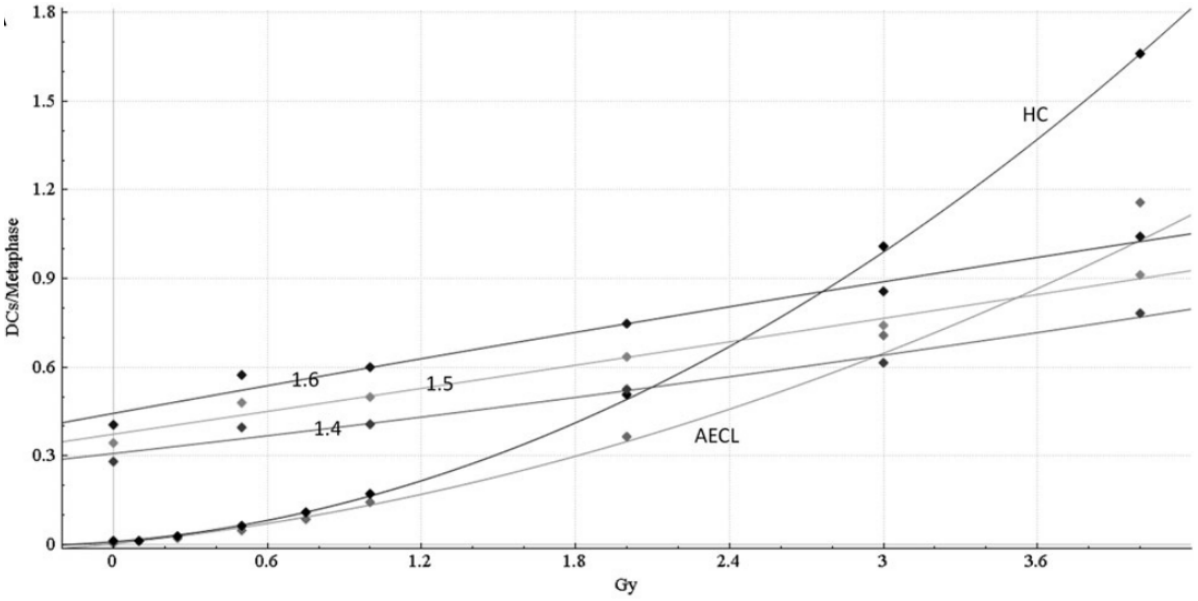


Figure 2.10 – Calibration curves of the 2016 version of ADCI introduced in Rogan *et al.* [88]. The two human expertises are provided by HC, CNL (plotted as AECL). Three calibration curves are plotted for ADCI, depending on the hyperparameter σ of the MC-DC classifier.

number of segmented objects is a very reliable feature for metaphase selection: if the number is too low, it may be because its chromosomes are overlapping, therefore reducing the number of detected objects. On the other hand, if the number of objects is too high, the image may contain chromosomes from two cells instead of one. The parameter σ of the improved SVM MC-DC classifier (which is the standard deviation of the radial basis function used as a kernel) introduced in [85] can be tuned by the user to balance specificity and sensitivity. If irradiation doses are known for a set of reference blood samples, ADCI has the ability to estimate a calibration curve. In [88], this is done in comparison to manual scoring provided by two labs, Health Canada (HC) and CNL (Canadian Nuclear Laboratories, formerly Atomic Energy of Canada Limited, or AECL). The calibration curve comparing ADCI to those laboratories is displayed in Figure 2.10.

Nevertheless, as shown in Figure 2.10, a large number of DC detections in this version of ADCI are false positives, which tends to skew calibration curves upward at the origin. While the aberration frequency for healthy patients (0 Gy) is around 10^{-3}, we can see in Figure 2.10 that ADCI estimates a much higher aberration frequency at 0 Gy. While it is possible to reduce that number of false positives by imposing stricter thresholding in the MC-DC classifier, this tends to skew the calibration further down for high doses. The

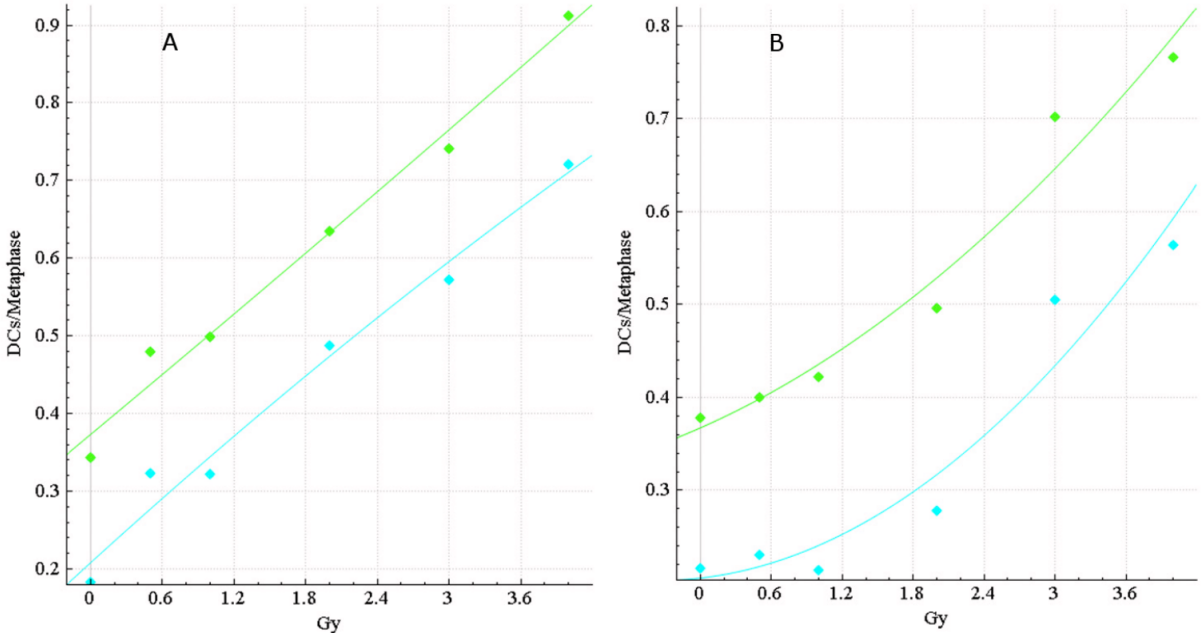


Figure 2.11 – Figure displaying the improvements brought by applying false positive filters implemented in the 2017 ADCI version introduced in Liu *et al.* in [84]. A) shows the calibration curve on a sample prepared by Health Canada (HC), while B) shows the same for a sample prepared by Canadian Nuclear Laboratories (CNL). Green curves show ADCI performance without false positive filters, while cyan curves show the performance after DC reclassification using false positive filters. (*Source*: [84])

following paragraph taken from [88] explains the relationship between false positives and dose estimation very well:

" *False positives have a proportionately larger impact at low doses, causing a stronger response in ADCI curves. At higher levels of radiation exposure, false positives tend to be balanced out by an increase in false negatives, i.e. DCs missed by the SVM. Beyond the level at which the number of false negatives exceed the number of false positives, the ADCI calibration curves exhibit a weaker response than manually derived curves.* "

Following versions of ADCI implemented changes that improved this false positive rate. The 2017 version of ADCI introduced in Liu *et al.* [84] introduced the MD-DC classifier improvements described in Section 2.4.2. Furthermore, metaphase quality has a large impact on MC-DC classification performance. Therefore, this version of ADCI provides improvements in metaphase selection. To adress this issue, a set of filters were designed to rank metaphases based on morphological features like average length-width ratio of chromosomes, centromere density (images containing closely clustered chromosomes are

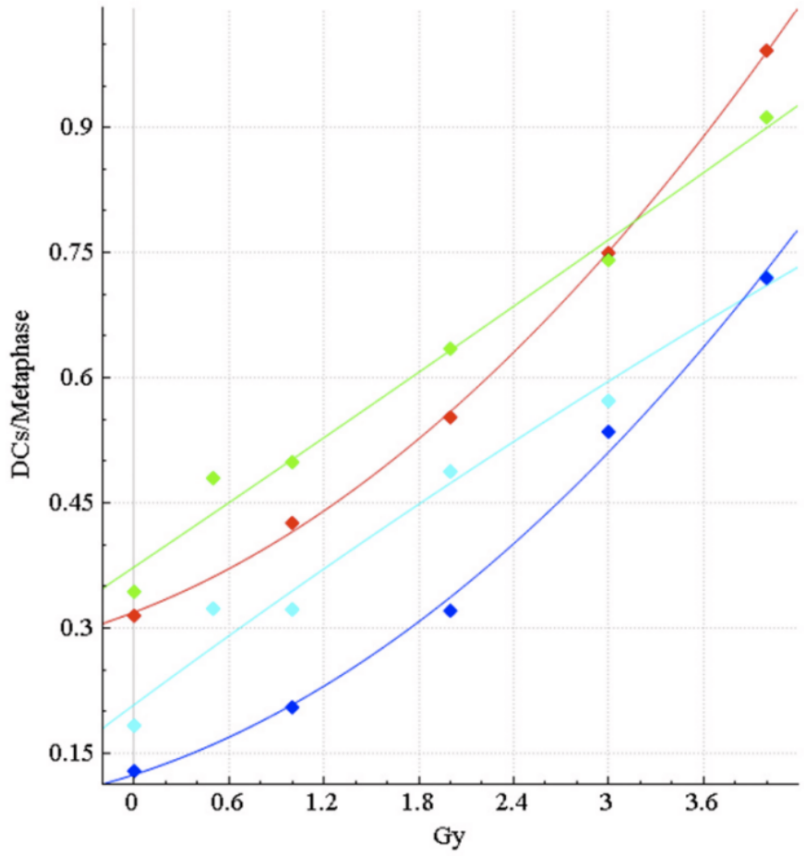


Figure 2.12 – Figure displaying the improvements brought by applying false positve filters and manually curating metaphases on the Health Canada (HC) sample. Although [84] implements metaphase selection algorithms, this figure displays the impact of manual metaphase selection. Green curve is evaluated on a un-curated dataset, without false positive filters. Red curve is curated, but estimated without false positive filters. Cyan curves uses false positive filters but is evaluated on an un-curated dataset. Finally, blue curve is evaluated on a curated dataset with false positive filters.

ranked lower), chromosome contour smoothness, total object count (metaphases with less than 35 objects or more than 50 are ranked lower), ratio of identified to unidentified objects, etc. A global image score was built as a linear combination of the previously mentioned image features. The weights of this linear combination were optimized with a grid search by minimizing the error to a known calibration curve. Finally, this weighted sum was used to rank images to be processed by ADCI. The improvements caused by re-classification of false positives can be seen in Figure 2.11. This figure also shows the relationship between sample provenance and calibration curve shape, as the radiation response of the Health Canada (HC) sample is almost linear, while the quadratic shape of

the calibration curve is much more visible for the Canadian Nuclear Laboratories (CNL) sample.

False positives reclassification and image selection lead to a very low level of false positives, and prevents the usual over-estimation of low doses in ADS systems. False positives can be further reduced by curating metaphases. The impact of this curation process can be seen in Figure 2.12.

the image selection models introduced in [84] rely on user-specified thresholds. In the 2019 version of ADCI introduced in [89], Li *et al.* demonstrate a procedure to find the parameters of those image selection models. A laboratory builds a calibration curve dataset comprised of a set of slides irradiated at different doses. As the ground-truth dose is known for each sample, it is possible to evaluate the dose estimation error. The parameters of the image selection models described in [84] are enumerated exhaustively, and the best model is chosen with respect to the calibration curve fitting error. Dose estimation is therefore completely automated, as long as a reference sample from a laboratory can be made available. On a laptop equipped with a GTX 960M, this implementation of ADCI can process around 100 metaphases per minute.

2.5 Deep learning for biological dosimetry

While all the solutions described in this section rely on conventional computer vision techniques, recent papers have used deep learning to solve the DC detection task at the core of biological dosimetry.

In [90], Jang *et al.* evaluate Faster R-CNN as a model for DC detection. First, metaphases are selected with a ResNet classifier, to reject images containing two metaphases, under-spread metaphases or metaphases with exposition issues. Once this image selection step is finished two Faster R-CNN-based models are trained: the Counting Network (CN) and Identification Network (IN). Dicentric yield is estimated using the number of chromosomes counted by the CN and the number of DC counted by the IN. Figure 2.13 shows the objects detected by the two models.

Both the Precision and Recall for the IN are around 90%, which means that around 10% of DC candidates are false positives, which is a relatively high false positive rate for biological dosimetry. Therefore, while the pipeline is accurate for high doses, the performance is unsatisfactory at low doses (below 1 Gy).

This pipeline is used to estimate a calibration curve on an independent sample (see

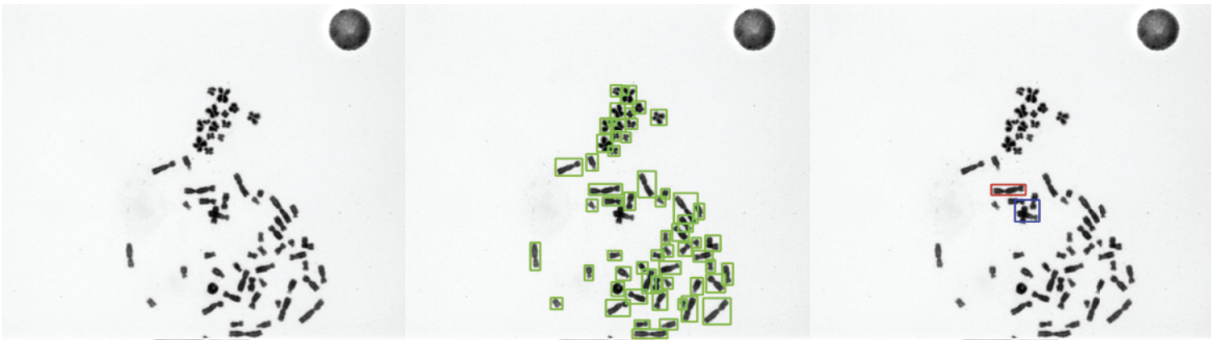


Figure 2.13 – Figure displaying the input image (left) detections of the Counting Network (CN) in the middle and Identification Network (IN), on the right.

Figure 2.14). Note that in this plot, the "dicentric rate" (and not the usual dicentric yield) is plotted as a function of dose. The dicentric rate corresponds to the probability for a single chromosome to be dicentric. The dicentric yield, on the other hand, is the average number of DC per metaphase. Assuming that every metaphase contains 46 chromosomes, the dicentric rate should be multiplied by 46 to get the usual dicentric yield. At 0 Gy, the model presented in [90] achieves a dicentric yield of around 0.01, which is approximatively 10 times higher than the manual dicentric yield (depending on laboratory counting protocol). Therefore, low doses are overestimated.

The approach proposed by Jang *et al.* partially circumvents the sophisticated metaphase selection algorithms presented in Section 2.4. Indeed, Jang *et al.* treat wether or not any chromosome is a DC as a Bernoulli-distributed random variable. Therefore, the radiation dose is linked to the mean of this distribution, instead of the average number of DC cell. Conventional calibration curves that link dicentric yield and dose rely on the implicit assumption that every metaphase contains 46 chromosomes which makes metaphase selection crucial. Incomplete metaphases quickly lead to an erroneous dicentric yield estimation: if an incomplete metaphase contains 39 chromosomes and 1 DC, a predicted DC count of 1 is likely to be biased downwards as the true (unknown) DC count can only match or exceed 1. In this scenario, Jang *et al.* effectively weight the DC count of 1 by a factor of $\frac{39}{46}$. As long as one can count chromosomes accurately and detect overlapped chromosomes estimating the dicentric rate is always possible even if some metaphases are incomplete.

Note that however, the training dataset of both neural networks relies on metaphases annotated by humans. As those metaphases were selected with human quality standards (good spread, object counts consistent with a single cell), some metaphase selection is still

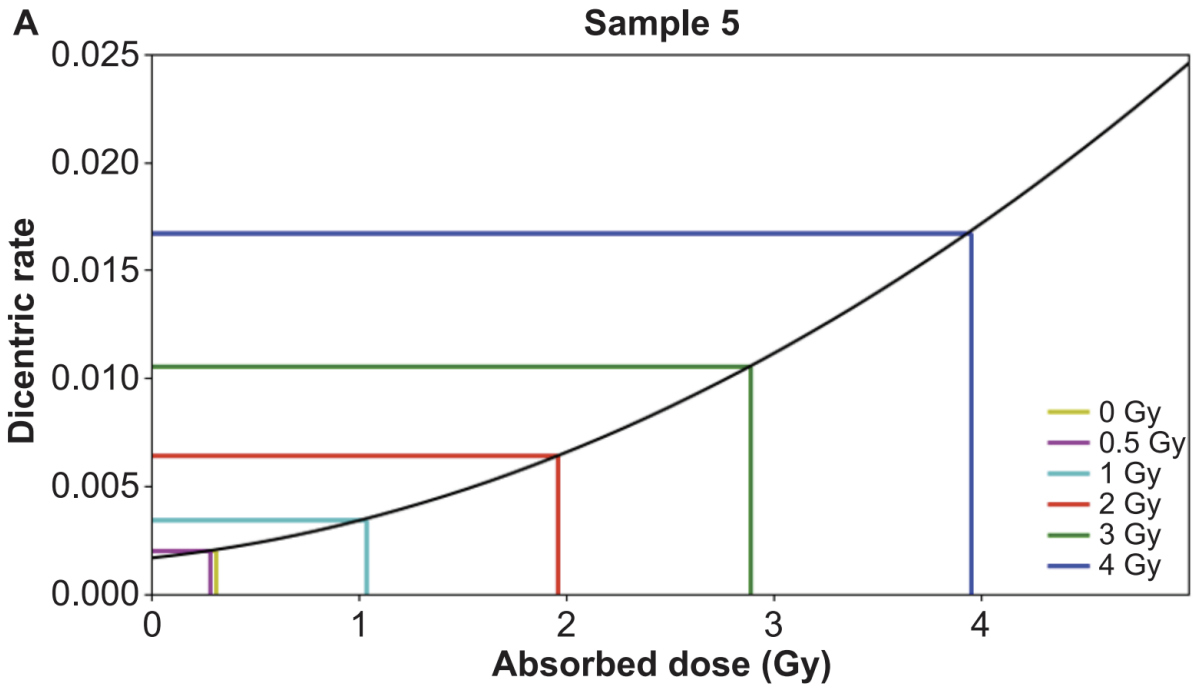


Figure 2.14 – Figure displaying the calibration curve estimated by the pipeline proposed by Jang *et al.* in 2021 in [90]. This curves displays dicentric rate instead of dicentric yield. Multiply dicentric rate by 46 (number of chromosomes in a normal metaphase) to get dicentric yield.

needed to ensure that the training and evaluation distribution match closely. However, this step is considerably simplified as only a ResNet-based binary classifier is needed in the proposed pipeline. Even if an incomplete metaphase is accepted the counting network weights the DC count accurately instead of completely rejecting the metaphase. This prevents the rejection of some DCs and improves recall which is especially relevant for low doses as the blood sample might only contain a single-digit number of DCs. Overall, the importance of metaphase selection is reduced but the step is not eliminated completely although it certainly leads one to wonder wether or not it would be feasible.

Another approach was proposed by Shen *et al.* in [91]. Instead of training a model in an end-to-end fashion to detect bounding boxes, chromosomes are detected with conventional computer vision techniques and then classified with a ConvNet. First, the metaphase image is segmented with K-Means. Chromosomes that are not adequately separated are identified based on their morphology, and individual chromosomes are separated with a watershed algorithm. Next, a binary classifier is trained with stochastic gradient descent to solve the MC-DC classification problem. The authors of [91] do not estimate a calibration

curve with their DC detection method which makes objective comparison with other approaches in this Chapter difficult.

2.6 DCScore: evaluating an ADS in a semi-automatic regime

2.6.1 Evaluating DCScore in realistic scenarios

ADCI aims to provide a fully-automated ADS. This is challenging as the number of false positives need to be controlled very tightly to prevent an overestimation of the dose. In a series of papers, researcher at LRAcc evaluated DCScore, a commercial ADS system provided by MetaSystems in a semi-automatic regime. An automated metaphase finder is used, and metaphases are selected by hand based on their quality. DCs are detected automatically and undergo a manual review by an expert. False positives are rejected by hand during this review. While this approach still need human experts, expertise time is reduced. It provides a good compromise between manual scoring (MS) and fully automated approaches like ADCI.

Within the context of a mass exposure where physical dosimetry information is not available, the IAEA established guidelines to ensure that victims receive adequate care based on the received dose [14]. Once a blood sample is taken and the cell culture step is done (as shown in Figure 2.2), an initial manual scoring of 50 metaphases (called the triage step) allows one to classify victims in range of doses. This first step is mostly useful to identify patients who received large doses. Once this is done, 500 images should be scored to refine the dose estimation. As manual scoring is very time-consuming, the triage step ensure that therapeutically relevant data is produced as soon as possible after exposure to radiation.

In real-world cases, the triage step has been shown to almost always underestimate the real dose as shown by Voisin *et al.* in 2001 [92]. ADS systems provide an alternative approach to this two step process, as the scoring time can be drastically reduced. In 2009, Vaurijoux *et al.* [93] compared the triage step to a different strategy where several hundreds of metaphases were scored using DCScore. The number of metaphases scored automatically was chosen so that the analysis time is roughly the same for the two strategies. Both metaphases and DCs were selected semi-automatically. In both cases, DCScore made suggestions, and a human operator performed a review. Vaurijoux *et al.* use real

data from 46 victims of the Dakar 2006 accident, where an iridium-192 source from a gammagraphy instrument was not properly returned to its shielded container. The comparison between the DCScore and manual scoring calibration curve is shown in Figure 2.15.

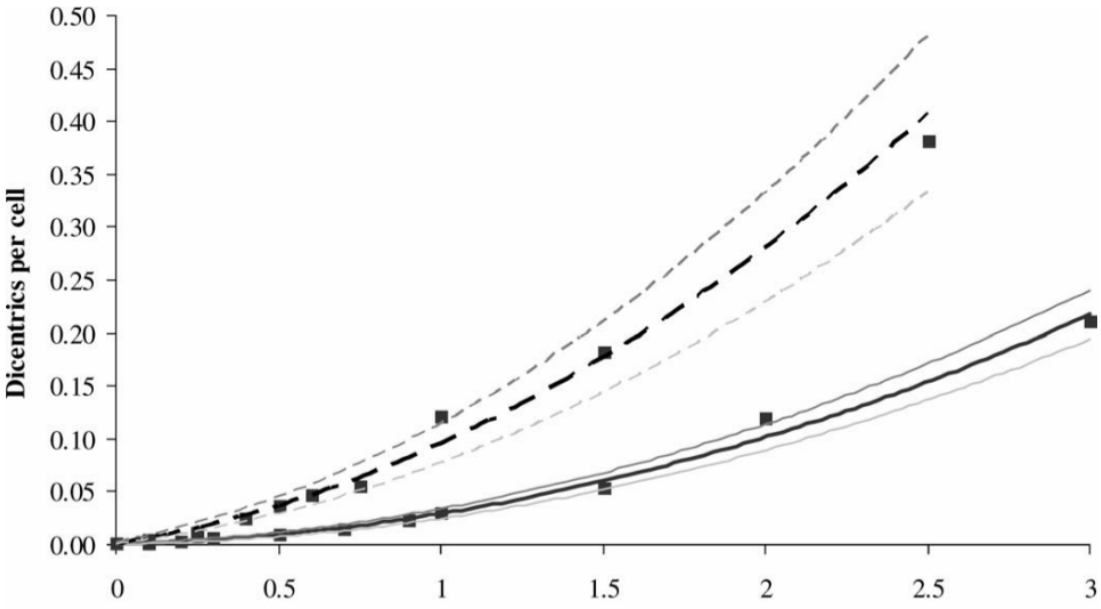


Figure 2.15 – Calibration curves for manual scoring (bold black dashed line) and semi-automated scoring (bold black line) with manual review to supress false positives. (*Source*: [93])

Note that this calibration curve showcases significant differences from calibration curves produced by other ADS systems like ADCI, as seen in Figure 2.10. This is because of the human review, which suppresses false positives very effectively. However, this leads to a systematic underestimation of doses. This is because the recall is low which leads to an underestimation of the true number of DCs.

Furthermore, the authors compare DCScore and manual scoring in their ability to classify patients in range of doses. First, they note that misclassification during triage is common with the 2006 Dakar data: 50% of patients were classified in a different dose range depending on wether scoring was performed on 50 metaphases of 250. While this may seem very high, this needs to be recontextualized as the dose range for the Dakar accident is low, and it is likely that triage would perform better in higher dose scenarios.

On the contrary, dose range classification of DCScore and manual scoring are in agree-

ment in 96% of cases. Like with ADCI, slide-to-slide quality variations are identified as a key factor of performance variation for DCScore. Again, shown in Figure 2.15 the low-dose context of the 2006 Dakar accident is where DCScore shows the best performance, i.e., the lowest difference to the manual scoring calibration curve.

In [93], expositions are assumed to be uniform. However, accidental expositions may also be partial. In the uniform case, the number of DCs follows a Poisson distribution, as explained in Section 2.1. However, in the partial exposition case, the blood sample is effectively a mixture of an unexposed sample and an exposed sample with an unknown mixing factor. The distribution of aberration count follows a Zero-Inflated (ZIF) distribution, where the probability of a cell containing zero aberration is much higher than in the Poisson case. Wether or not DCScore can provide accurate triage in the case of partial expositions was studied by Vaurijoux *et al.* [94] in 2012 using in vitro experiments. Irradiated and un-irradiated samples are mixed in different proportions and the ability of DCScore to provide an accurate dose estimation in this challenging context is evaluated. The authors again observe high agreement between manual scoring results on 500 metaphases and ADS. Furthermore, as DCScore lets the authors process a much larger number of cells than MS, they found that "*ADS is able to detect a dose significantly different from zero in samples having a lower fraction of irradiated cells compared to the MS method*".

The results of [94] were refined in 2013 by Gruel *et al.* in [95] where a large scale emergency is simulated in vitro. The authors provide finer bounds on the number of cells needed for accurate triage with DCScore compared to manual scoring. As the suspected dose gets lower, the number of metaphases to score in order to accurately estimate the dose should be larger. For doses above 1 Gy, 1500 images scored with DCScore are sufficient, but lower doses especially in the context of a partial exposition may require 3000 images to be accurately estimated. Results from several european laboratories using this semi-automated approach for large scale expositions were compared in 2013 by Romm *et al.* [96]. The overall biological dosimetry pipeline (from blood sampling to dose estimation) was found to be sufficiently normalized, so that the difference in calibration curve estimation between laboratories was minimal.

2.6.2 Exploring DCScore shortcomings

Metafer has several tunable parameters for metaphase discovery, like microscope scanning speed and detector sensitivity. In this section, we discuss two recent papers (2019 and

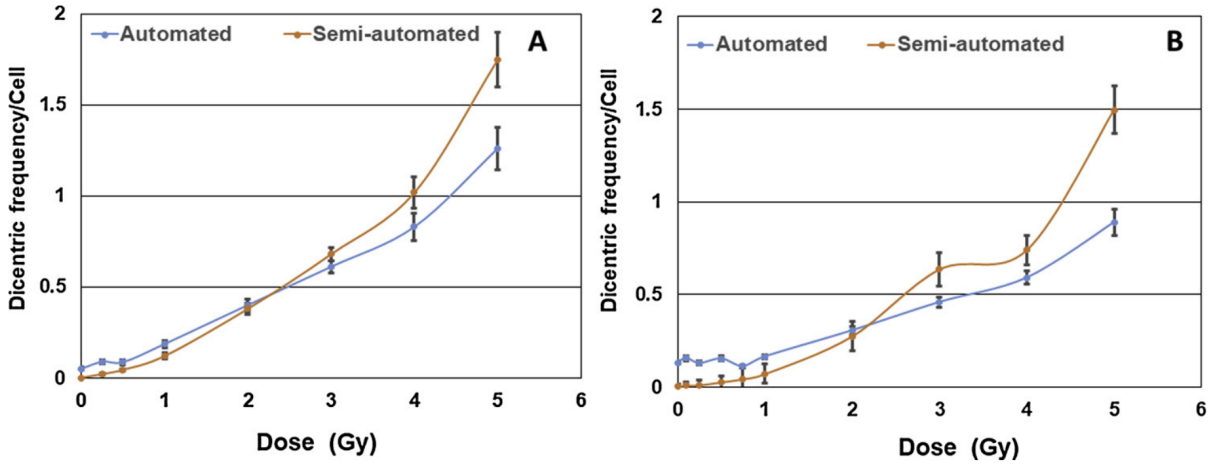


Figure 2.16 – Calibration curves for automated (blue) and semi-automated (orange) DCScore. Left panel shows calibration curve for X-rays, right panel for γ-rays. Note that the authors did not fit a linear-quadratic model to the dicentric yield point cloud, which explains the unusual look of the calibration curve compared to other examples in this thesis. (*Source*: [97])

2020) exploring the relationship between metaphase selection performance and accurate dose estimation. In [97], Ryan *et al.* evaluated the relationship between those parameters and dose estimation accuracy. Furthermore, they compared automated DC detection with semi-automated detection. They found that for high doses (between 1 and 3 Gy), DCScore can accurately estimate doses in an automated fashion, and is good enough for preliminary triage of patients. Furthermore, manual review of metaphase images does not improve performance. Like previous papers [93]–[95], they find that the false positive rate is too high for accurate low dose estimation, and that high doses tend to be underestimated because of the insufficient recall. This pattern is very clear in Figure 2.16.

The performance of DCScore was found to depend strongly on the accurate rejection of unusable images (images containing several cells, underspread cells or poorly exposed cells). This is because dicentric yield estimation relies on the assumption that metaphases are complete, i.e they contain all 23 pairs of chromosomes. However, automatically detected metaphases may be incomplete because of failures in the metaphase detection software. A lower number of chromosomes reduces the actual probability of observing a DC in a specific cell, which means that a larger number of cells does not contain any dicentric. This can be wrongly interpreted as a partial exposition. In [98], Endesfelder *et al.* use the chromosome count provided by DCScore to adjust the uncertainty around the dose estimated by DCScore. This strategy is analoguous to the one used by [90] and de-

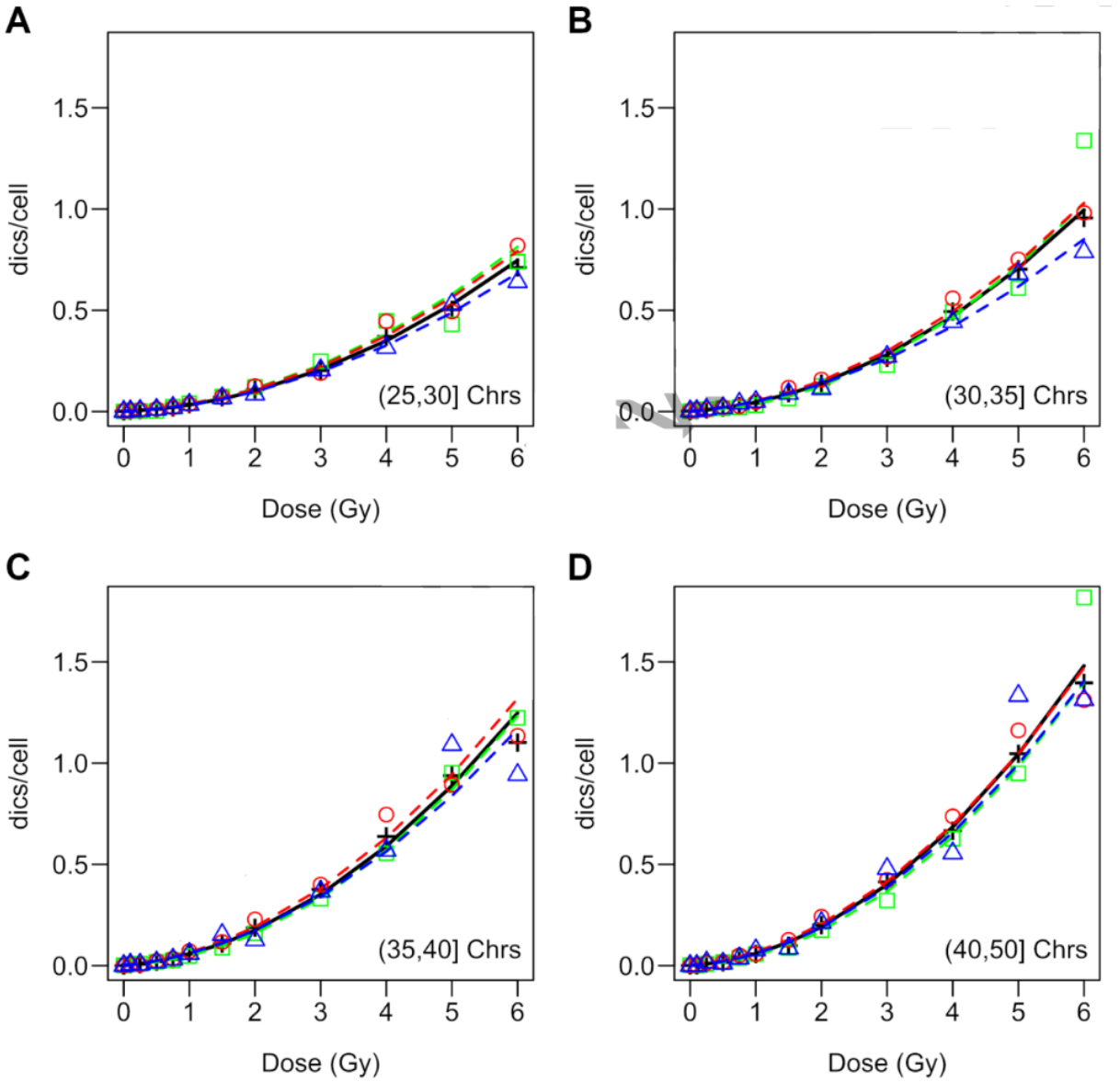


Figure 2.17 – Figure showing the impact of chromosome count on calibration curves. [98] uses three different blood samples, shown respectively with blue, red and green calibration curves. Black calibration curve corresponds to pooled dataset. Chromosome counts are shown in lower right corner of all quadrants. (*Source*: [98])

scribed in Section 2.5. The authors confirm that overdispersion is present (mean is greater than variance) at high doses, suggesting that correcting dicentric yield estimation with chromosome count is necessary. Excluding metaphases with low chromosome counts tends to reduce overdispersion. This effect is shown in Figure 2.17.

2.7 Conclusion

In biological dosimetry, a radiation dose is inferred by estimating a dicentric yield. This yield corresponds to a proportion of DCs per cell. Accurately identifying DCs is a challenging computer vision problem, which has received significant attention over the years, even going back to the late 1980s.

Those ADS systems relied on a pipeline of object detection and object classification. DCs are identified based on their number of centromeres, which are identified by counting constrictions in chromosomes width profile . The Edinburgh dicentric hunter is an example of those early ADS systems [70]. Over the years improvements in computer vision and pattern recognition led to better performing ADS systems. The ADCI is the leading example for fully automated ADS systems, and reiceved continous upgrades over 7 years, culminating in Li *et al.* [89], which was published in 2019. Another example is DCScore, which was not described as precisely as ADCI. However, its potential as an ADS system in emergency situations was assesed by several differents labs, as described in Section 2.6.

All of the ADS examples presented in this section are pipelines of processing step. Object detection and classification are treated in consecutive, separate step. On one hand, this offers unparalleled granular control over every step. This makes each step interpretable, and it is usually very easy to know why a specific object is misclassified from its morphological features. On the other hand, a considerable amount of design work was needed. Furthermore, errors in pipeline tend to compound: a segmentation error will often lead to a classification error, which could lead to a dose estimation error. Therefore, each step must be highly reliable, as the performance of every algorithm is strongly correlated.

While DCScore does not incorporate sophisticated metaphase selection algorithms for fully automated operations, it has been shown to be very effective assistants to human experts. In a series of papers published by LRAcc researchers [93]–[95] it was shown that human post-processing of DCScore detections could provide usable dose estimation faster than completely manual scoring. However manual scoring is more accurate, especially for low doses. This remains true for challenging contexts like partial exposition detection.

More recently, several papers investigated ConvNets for DC detection. In a 2023 paper [91], Shen *et al.* propose a conventional pipeline where chromosomes are first detected and then classified. The binary MC-DC classifier is a ConvNet trained with SGD. While the authors demonstrate high levels of precision and recall they do not present any calibration curve estimate. In a 2021 paper [90] Jang *et al.* trained Faster R-CNN to detect DCs.

While annotating metaphases with bounding boxes is costly, this removes the need for a sophisticated chromosome detection step.

Chapter 3

Two-stage chromosomal aberration detection with patch classification

In this chapter, we work with a dataset of chromosome patches as a preliminary study. Building such a patch classification dataset is much easier than the spatially labelled datasets used later in this thesis. However, as we will see below, building components of an ADS system using only this dataset is very challenging. We focus on several different potential uses for this dataset within the context of an ADS system.

First, we investigated the performance of deep classifiers, and compare it to conventional methods. While reaching a high level of performance (even in a semi-supervised regime) on this classification dataset is relatively easy, it does not transfer to real metaphase images. We provide potential explanations for this fact.

If we can learn to synthesize chromosomes in an unsupervised fashion, we would be able to produce synthetic, annotated metaphases. This could be used to pre-train object detection models. We investigates two chromosome simulation strategies, VAEs and a specific GAN architecture called pix2pix (see Section 1.2.1). Both of those strategies struggle with synthesizing diverse and realistic chromosomes without artifacts.

Finally, given our lack of success in synthesizing satisfying chromosomes, we build a synthetic annotated metaphase dataset from chromosome "skeletons". Using this dataset, we demonstrate that model performance depends on the number of images in the training set, and on the dose. Models trained on one given dose usually do not reach the same level of performance when evaluated on a different dose.

3.1 Datasets

3.1.1 Patch dataset

Our dataset is comprised of 17061 grayscale chromosome patches of varying sizes. This dataset contains two classes, MCs and DCs. 6349 of those chromosomes are DCs, while 10712 are MCs. In the remainder of this chapter, we denote this dataset as $\mathcal{D} = \{(x_1, y_1), \cdots, (x_N, y_N)\}$, where N is the number of images. $\mathbf{X} = \{x_1, \cdots, x_N\}$ is the set of input image $\mathbf{y} = \{y_0, \cdots, y_N\}$ is the set of labels, with $y_k \in \{0, 1\} \, \forall \, 1 \leq k \leq N$. $y_k = 0$ if the k-th image is a MC, and 1 if it is a DC

The chromosome patches underwent a significant amount of pre-processing. First, the images where segmented with K-Means to separate background and foreground. K-Means was chosen instead of a global thresholding method because it dealt with the intensity differences of chromosomes better. For example, the region between sister chromatides usually exhibit higher intensity, and Otsu thresholding produces noisy segmentations in this case. Once this step was completed, secondary object are rejected with connected components, on the basis of their size. We keep the largest object in the patch, assuming that it was the chromosome object. Finally, the image intensity values are normalized between 0 and 1 in the following fashion:

$$\tilde{x} = \frac{x - \min(x)}{\max(x) - \min(x)}, \tag{3.1}$$

where $\tilde{x}$ is the normalized image.

3.1.2 Skeleton datasets

Using the aforementioned patch dataset, we built a "skeleton" dataset. First, for every chromosome, we extracted the centerline, and computed the distance between the boundary and the centerline. Centromeres were identified by locating minimas of this distance. A stylized chromosome representation was built using this information. A grid of chromosomes using this skeleton representation is shown in Figure 3.2. To ensure that every image has the same size,they are padded to a size of 128 by 128 pixels, i.e., $x \in [0, 1]^{H \times W}$, with $H = W = 128$. A grid of padded chromosome images is displayed in Figure 3.1. We also built a labelled metaphase datasets using the chromosomes skeletons introduced above. First, we extracted the locations of chromosomes in real metaphase images. Then,

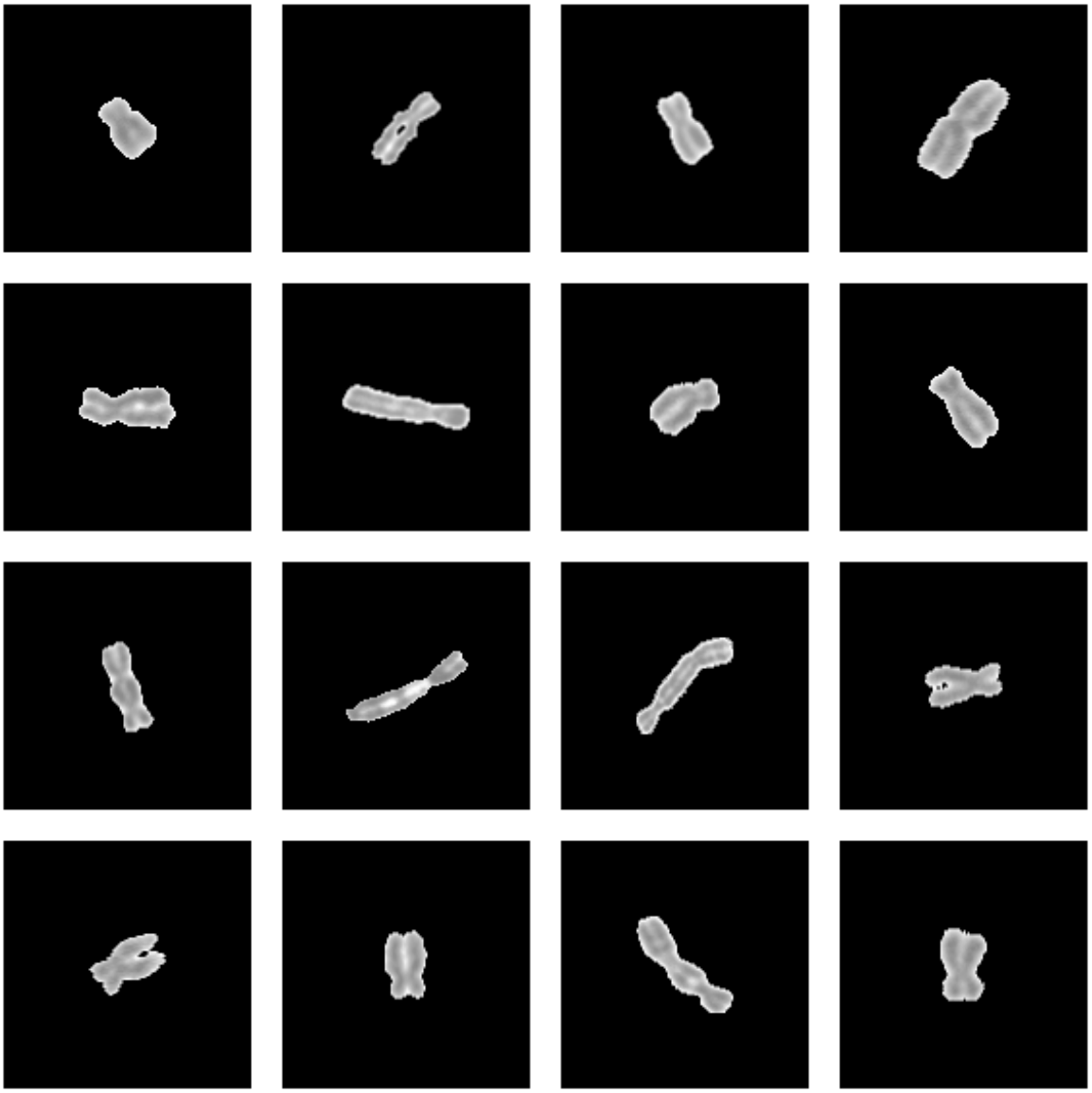


Figure 3.1 – A grid of padded chromosome images.

we sampled from our skeleton dataset at random and we overlaid those skeletons on a white background, at the locations extracted previously. Chromosomes skeletons where randomly rotated and resized, to increase metaphase diversity. Finally, we sampled a realistic number of DCs using the manual calibration curve established by the LRAcc (see Chapter 2). We built 3 different datasets at dose levels of 1 Gy, 2 Gy and 5 Gy. For every dose level, we retrieved the corresponding average number of DC per metaphase using

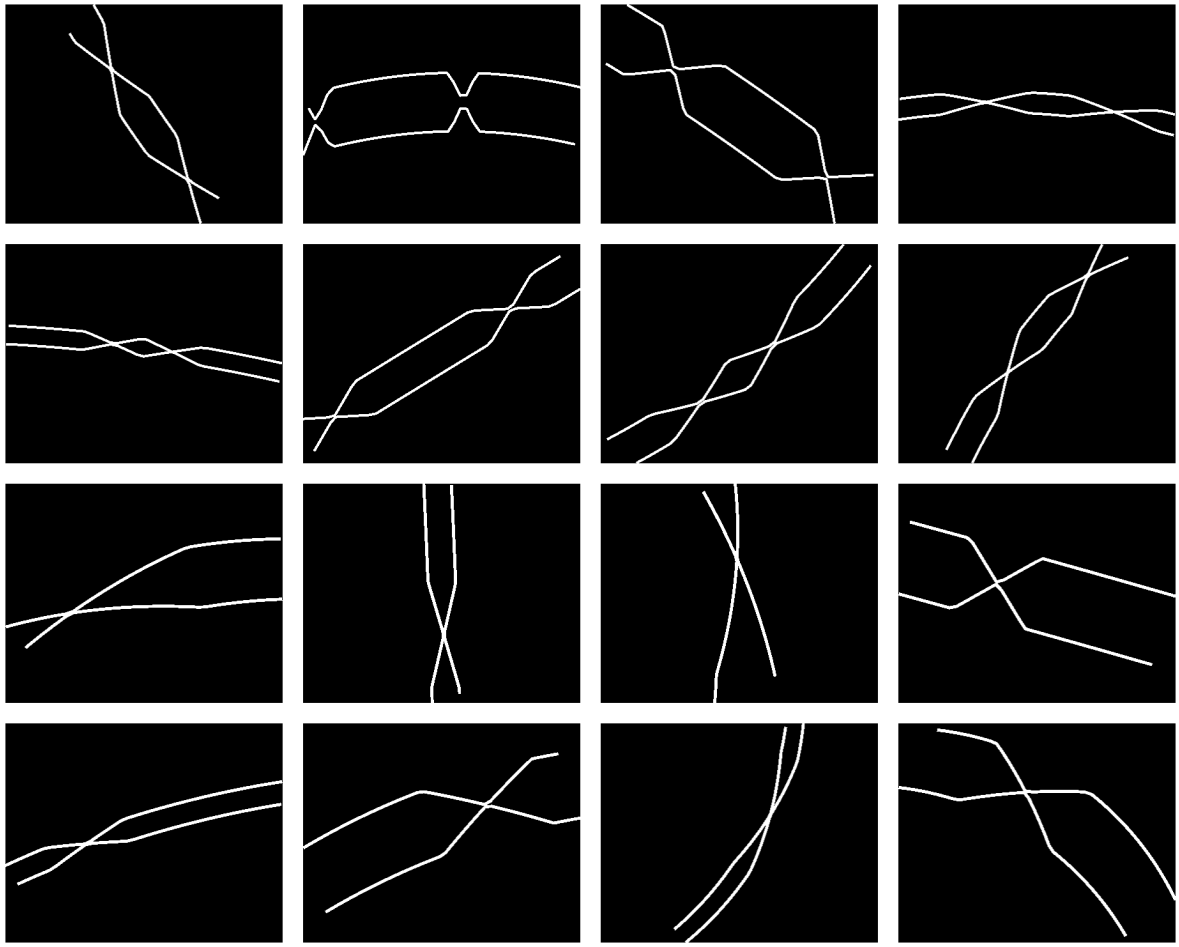


Figure 3.2 – A grid of samples of chromosome skeletons.

the manual calibration curve, and sampled a Poisson distribution with this mean to get a DC count per metaphase. In our first experiments with Faster R-CNN, we quickly found that if we labelled every MC, the training of the model falls into a local minima where minority classes are never predicted. Therefore, we only labelled chromosome aberrations and MCs are effectively treated as background. Figure 3.3 gives an example of cropped training image. For each dose, we built a test dataset of 5000 images.

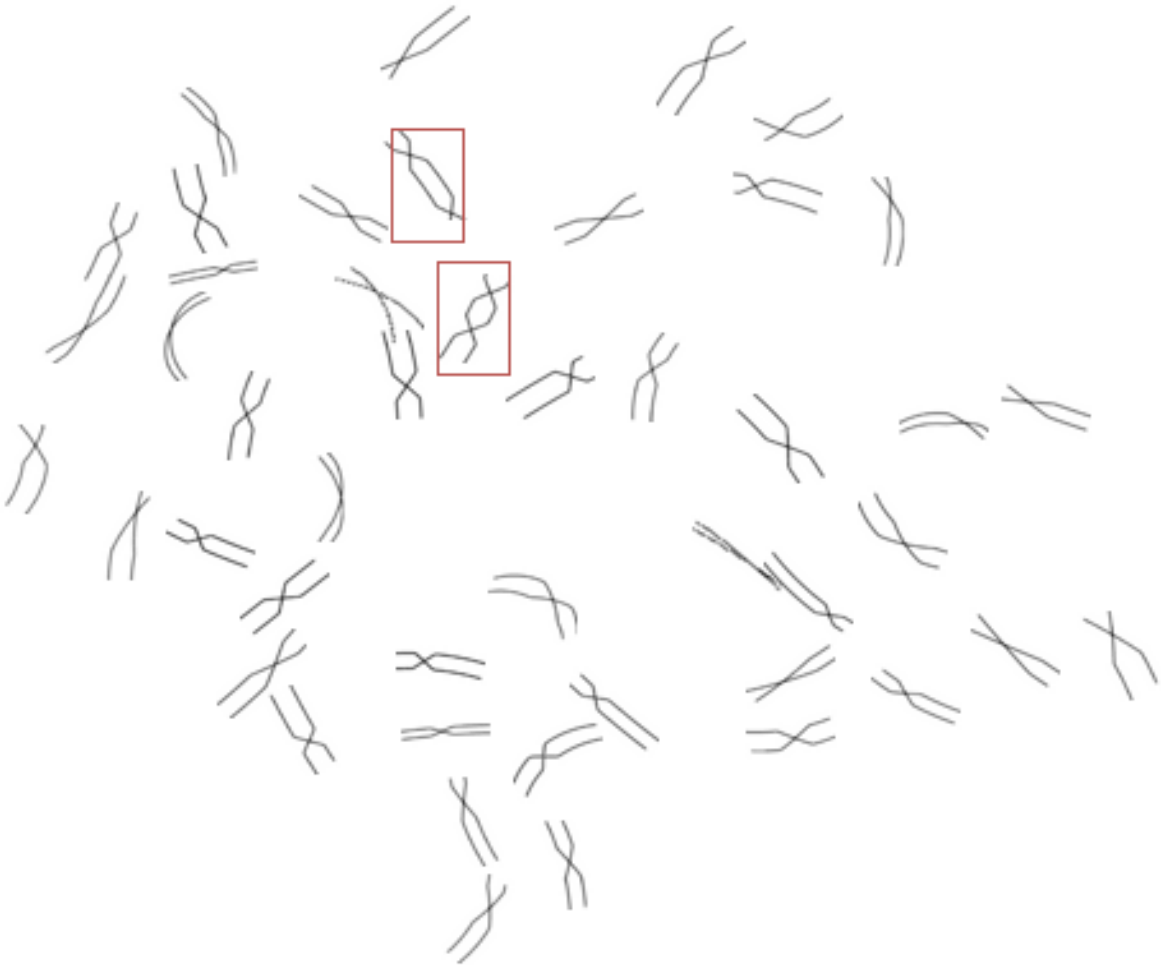


Figure 3.3 – Cropped image of the synthetic dataset. The two DCs in this image are indicated by red bounding boxes.

3.2 Chromosome classification

3.2.1 Resnet-based classifier

To establish a DNN-based, fully supervised chromosome classification baseline, we used a ResNet model [8]. Our goal was to demonstrate that conventional DNN-based classifiers could reach a high level of performance. This model is trained with Adam [12] (a variant of SGD), with a learning rate of 0.0001, on the dataset $\mathcal{D}$, as described in Section 3.1.1. We did not use any data augmentation or learning rate scheduling. Furthermore, we did not perform any architecture search or hyperparameter tuning. Therefore, we do not use

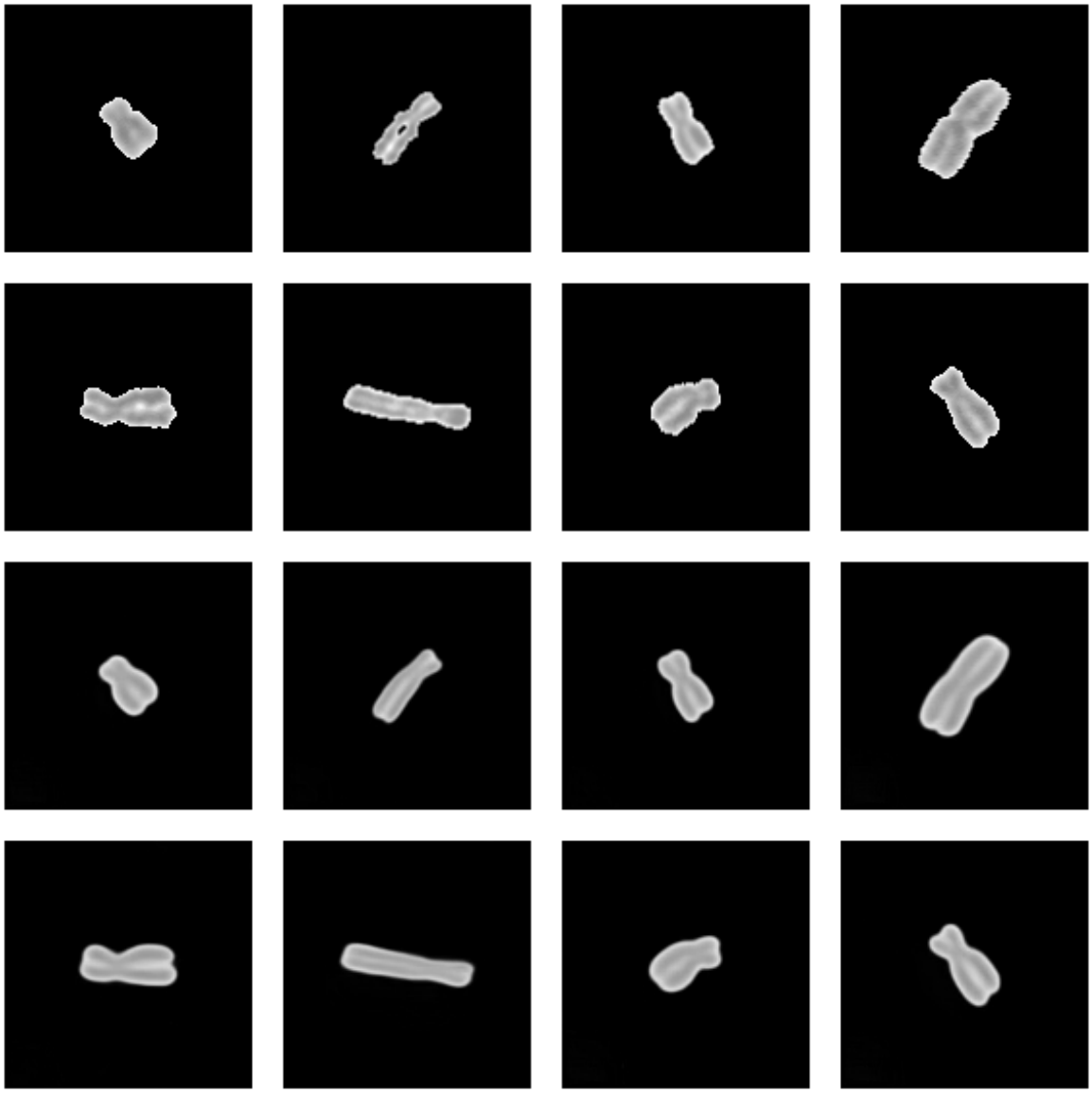


Figure 3.4 – Reconstruction grid of unseen chromosome patches. The first 2 rows are unseen input images, the last 2 rows show their reconstruction by the VAE. Notice that the reconstructions are smoother than the inputs. While the L_2 criterion is a common reason for this effect, the output of the same model trained with an L_1 criterion are very similar.

any validation set, and the performance was evaluated on a test set comprised of 20% of the training data. The training dataset is comprised of the remaining 80 % of $\mathcal{D}$.

3.2.2 Chromosome patches autoencoder

We trained a convolutional VAE as a simple unsupervised feature learning technique. This model is trained on the dataset $\mathbf{X}$ with an L_2 reconstruction criterion and a KL weighting parameter of 1 (see Chapter 1, Section 1.2.1 for an overview of VAEs). We chose a latent space dimension of 256. In Figure 3.4, we show some examples of patch reconstruction on unseen data from this VAE.

3.2.3 Latent space classifier

The trained VAE model mentioned in Section 3.2.2 provides a way to map image samples into a lower dimensional latent space of dimension d. We set $d = 256$ in our experiments. Knowing the image labels associated with each low-dimensional feature vector, we can train simple binary classifiers, like a logistic regression model. Again, the goal here was not to demonstrate the highest level of performance possible. Other methods could have been studied, like gradient boosting, random forests or SVMs. Because of the high number of latent vectors, this logistic classifier was trained with a batch stochastic gradient method called SAGA [99]. We used the `scikit-learn` [100] implementation of this algorithm in our experiments.

3.2.4 Performance results and commentary

First, we computed the performance of our ResNet classifier baseline in a conventional fashion, with a test set comprised of 20% of overall images. Then we explored the potential of classifiers trained on the embeddings of a variational autoencoder. Our goal was to evaluate the potential of a logistic regression classifier trained on a small number of embeddings from a VAE. If this strategy succeeded it would be possible to train an effective MC-DC classifier using a large, unlabelled dataset of patches (to train a VAE) and a few labelled patches. Therefore, we intentionally made the training set of our logistic classifier small (at most 10% of the complete dataset). This classifier takes a VAE embedding as input and outputs a probability distribution over both classes. Furthermore, we also wanted to evaluate wether our logistic classifier was sensitive to shifts in the distribution of its training data.

Therefore, we built k non-overlapping splits of the dataset $\mathcal{D}$. Each one of those splits S_k is composed of image-label pairs, i.e $S_k = \{\{\mathbf{X}_k^T, \mathbf{y}_k^T\}, \{\mathbf{X}_k^E, \mathbf{y}_k^E\}\}$, where $\mathbf{X}_k^T \in \mathbb{R}^{n_k^T \times d}$ where d is the latent dimension of the autoencoder, so that $n_k^T + n_k^E = N$. Here, $\{\mathbf{X}_k^T, \mathbf{y}_k^T\}$

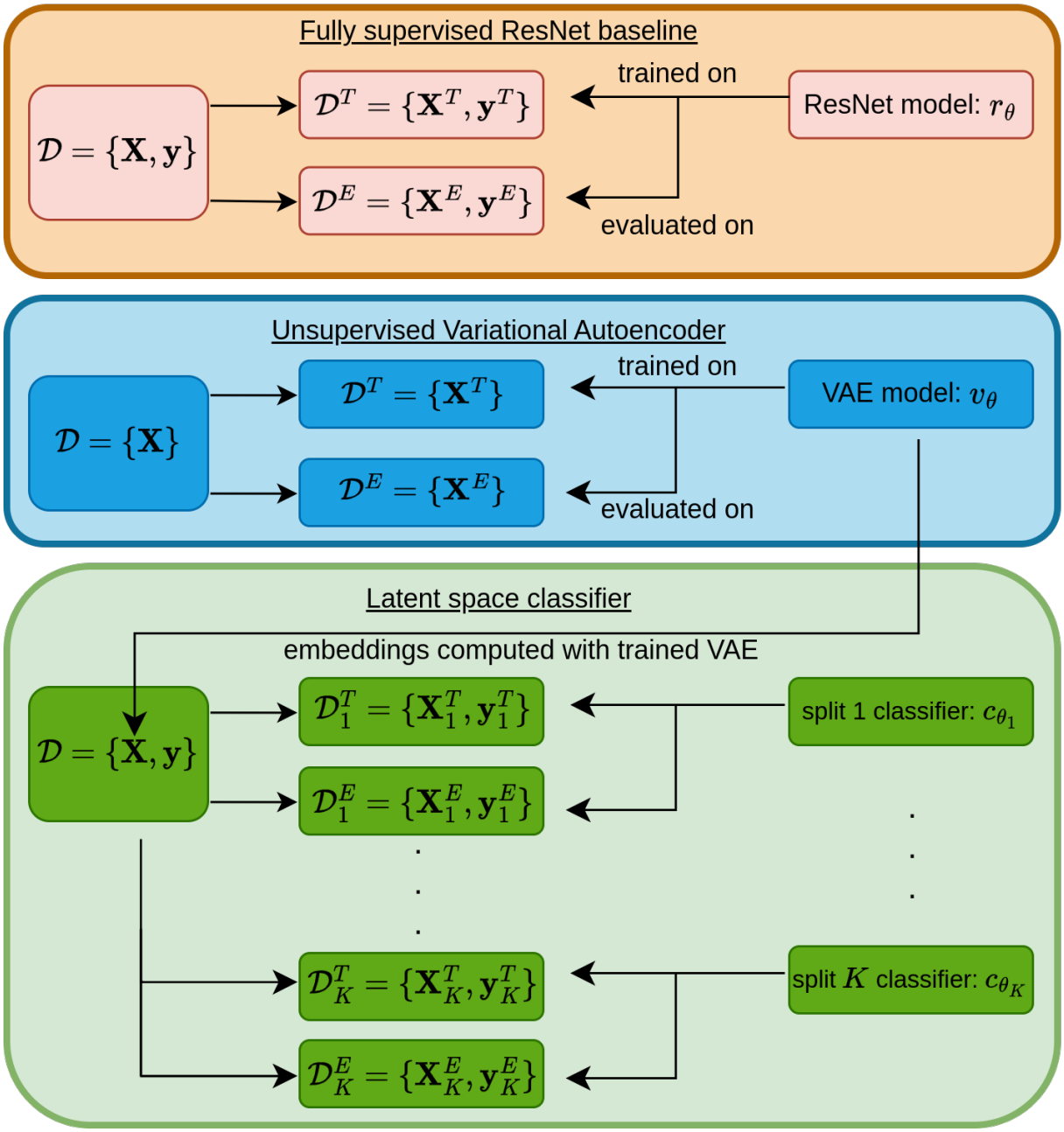


Figure 3.5 – Description of our weakly-supervised experimental setup. The fully supervised ResNet baseline is trained on a simple train-test split. The VAE is trained in an unsupervised fashion (it does not use $\mathbf{y}$) and its performance is evaluated on a test split. Once the VAE is trained its encoder is used to compute the latent repreentation for all images in $\mathbf{X}$. We build K different splits of this "embedded" dataset. Here, the training dataset is always less than 10% of the complete dataset. Performance is evaluated on the test set. As we have K different splits, the mean and variance of performance can be computed, to evaluate the sensitivity of the logistic classifier to data sampling. Results are visible in Figure 3.6.

is a training set while $\{\mathbf{X}_k^E, \mathbf{y}_k^E\}$ is a test set. Finally, we have $\mathbf{X}_k^T \bigcap \mathbf{X}_k^E = \varnothing \, \forall k$. Figure 3.6 shows the results of this evaluation.

The highest performance model is the fully-supervised ResNet baseline, reaching an AUC of 0.98. However, it is possible to reach a very high level of performance with much less labelled data and a strong feature learner, as the solid blue, purple and green lines show. We trained a VAE on $\mathbf{X}$, and computed the latent representation of every image in $\mathbf{X}$. We then used the embeddings and the corresponding labels to train classifiers. Those classifiers are trained on respectively 10, 5 and 1% of the training data, and evaluated on the rest. One can get over 90% of the performance of the supervised baseline with only 10% of the labelled data ! The non-linear embedding provided by the VAE leads to higher downstream performance than the linear embedding of PCA, as the dashed curves show. Furthermore, there is more performance variation in the downstream models that use PCA embeddings. While this level of performance might lead us to think that "two-stage" models which successively detect and classify chromosomes might be very simple to build, in practice we did not succeed in reproducing this high level of performance with patches extracted from metaphases.

The first reason we identified is the lack of representativity. In our dataset, all monocentric chromosomes correspond to chromosomes detected by DCScore (See section 2.6 for a review of papers examinating DCScore performance). We used a first patch dataset where the DCs were also extracted by DCScore. However, DCScore is not able to recover all DCs. For example, in some intensity conditions, the detection performance of DCScore worsens, which means that especially bright or dark DCs are not present in the dataset. This led to a phenomenon analoguous to label leakage; it became possible to separate MCs and DCs with simple thresholds on the mean and variance of the patch intensity. We corrected this problem by enlarging the dataset with manually detected DCs, to build a more representative dataset. However, we still observed difference in validation performance between the two subset of our dataset, depending on wether the DCs were detected by DCScore or a human expert.

We identified another explanation for this low level of DC detection performance. The performance of the detection model and the classifier are strongly linked. For example, if the second centromere of a DC is very close to the end of this chromosome, slight segmentation errors might lead to a bounding box that does not include this second centromere. In this case, there is no chance of accurately classifying this chromosome as a DC. In the same way, rejecting overlapping chromosomes is essential to ensure a high level

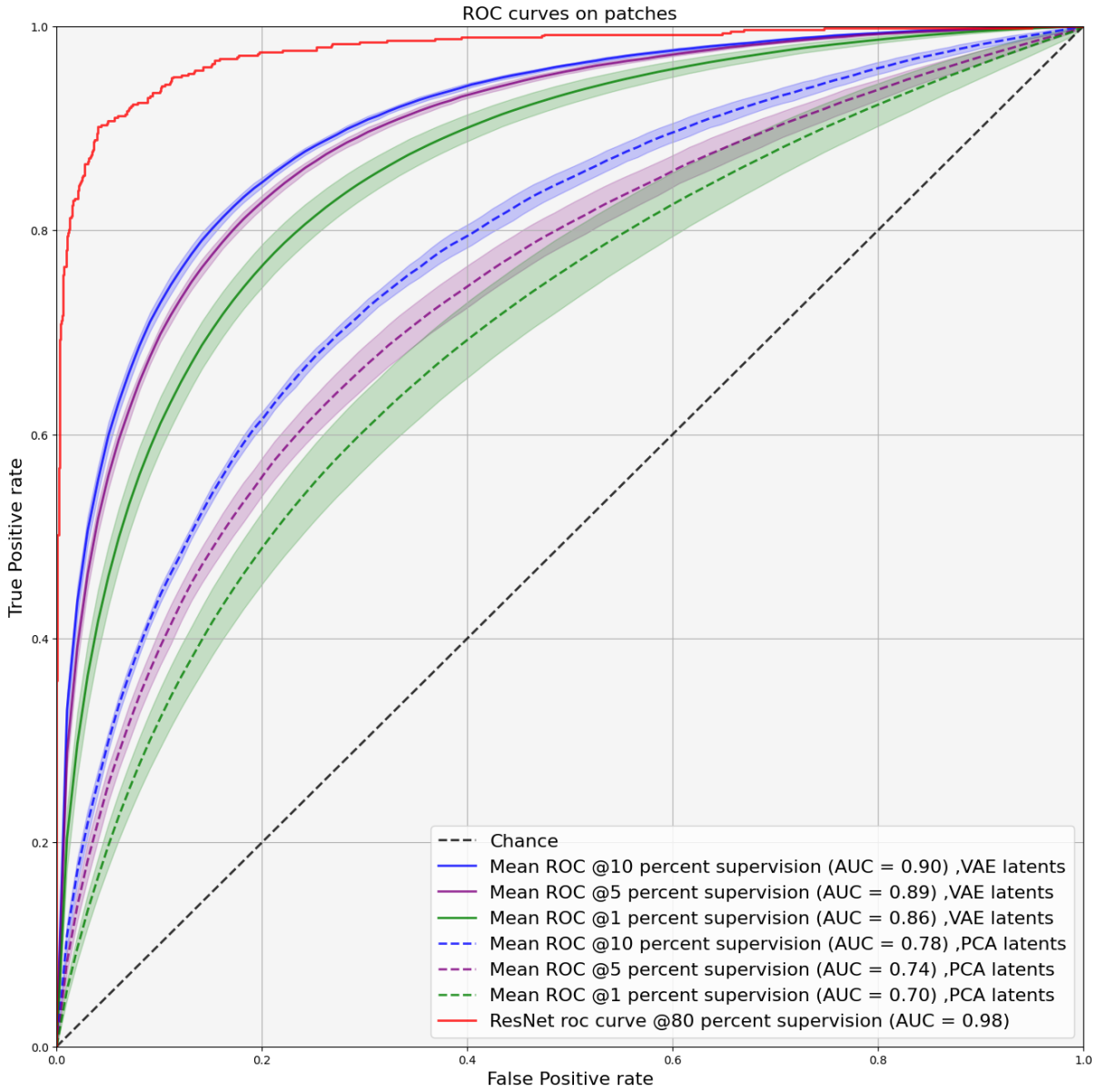


Figure 3.6 – Figure comparing the performance of several models. The dashed redline indicates the performance of the ResNet supervised baseline. The solid blue line indicates the mean performance of a logistic classifier trained on VAE latent features, if 10% of the data is labelled. The purple and green lines depict the performance of the same model, for supervision rates of 5% and 1% respectively. The shaded area shows performance variation across all splits (plus / minus one standard deviation). Finally, the dashed lines depicts the same performance for PCA embeddings, instead of VAE.

of performance for this class of "pipeline" DCs counters. Section 2.4 summarizes the level of effort required to build an ADS that relies on conventional computer vision techniques.

DL-based object detection models like Faster R-CNN [9] might require less design work and perform better, but large labelled datasets are required. Therefore, the next section investigates chromosome generation to synthesize labelled datasets.

3.3 Simulation of chromosome patches

Investigating chromosome generation is appealing for two main reasons. First, for VAEs, investigating links between latent variables and morphological variations might help one to build a better classifier. Instead of relying on a logistic regression, simple functions of some latent variables might be enough for classification. Second, a good chromosome simulator could be used to generate fully annotated metaphase images, which could be used to pre-train aberration detection models, or evaluate architectures on artifical data.

3.3.1 Simulating chromosomes with VAEs

Simulating chromosomes with a VAE can be done by sampling the latent space prior (which is a Gaussian with zero mean and unit variance), and decoding those samples into image space. While this is what the authors of [36] suggest, there are several caveats with this approach. First, as pointed in the introduction of [44], the true posterior over the latent variables rarely matches the prior. Therefore, sampling from this unit variance Gaussian prior might lead to low quality samples. To circumvent this issue, we investigate two alternative methods to sample from this posterior.

First, we projected latent space samples in an orthogonal basis, using PCA. We estimated the probability distribution of PCA loadings for every coordinate of the latent space using a Kernel Density Estimation (KDE). We estimate the bandwidth using the cross-validation algorithm implemented in [100]. A grid of chromosome samples is displayed in Figure 3.7 a). Those samples show low diversity, and are not very realistic. As the dimension of the latent space is quite high (256), density estimation methods struggle because of the high distance between samples. Therefore, we fitted a Gaussian Mixture Model (GMM) on the latent space of the VAE. Again, we sampled from this GMM, and used those samples as an input for the decoder of a pre-trained VAE. A grid of samples is shown in Figure 3.7. This improves samples quality and diversity, but introduces arti-

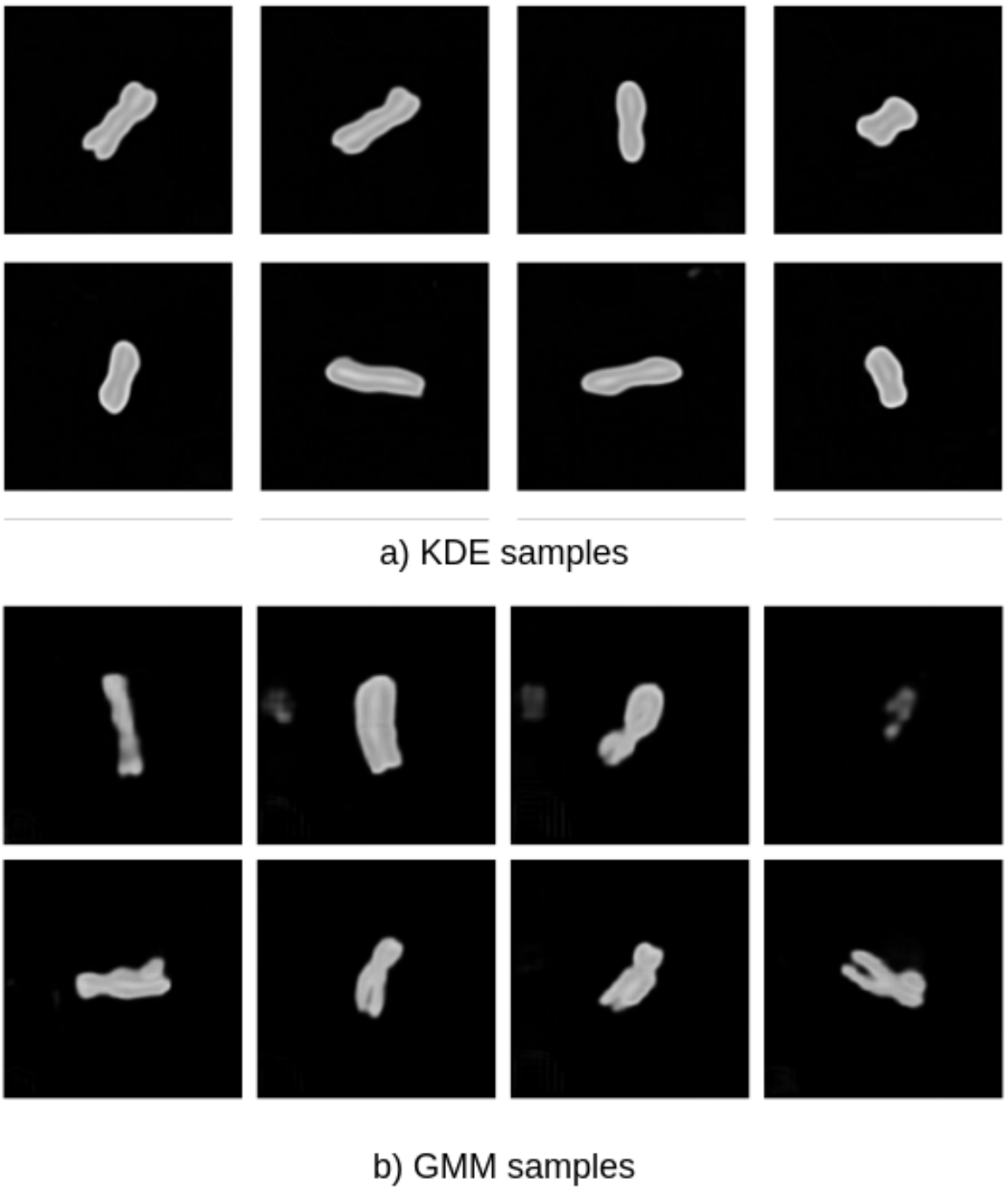


Figure 3.7 – a): Samples of the latent distribution estimated with KDE. b): Samples of the latent distribution estimated with a Gaussian Mixture Model. In both cases, those samples were decoded with the decoder of the VAE trained earlier.

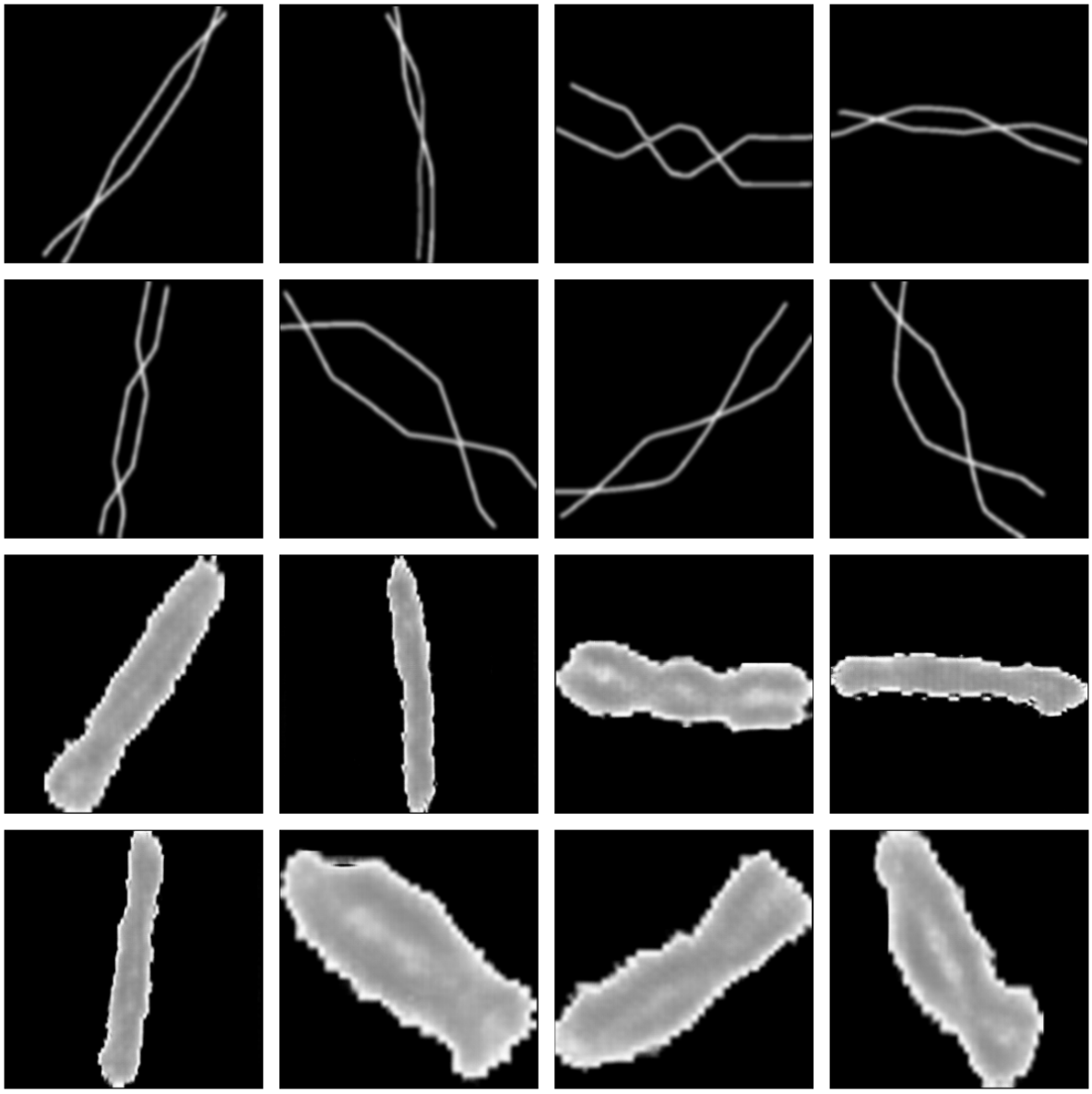


Figure 3.8 – 8 DC samples from pix2pix. Top 4 rows are "skeleton" inputs, bottom 4 rows are chromosome outputs.

facts. Overall, we found generating samples in a completely unsupervised fashion to be very difficult.

3.3.2 Simulating chromosomes with pix2pix

We use the pix2pix architecture [43] to synthesize chromosomes as illustrated by Figure 3.8 which shows a grid of samples from pix2pix. While chromosome samples have no artifacts, distinguishing MC and DC chromosomes remains difficult. Thin chromosomes often have completely indistinguishable chromatides which makes telling wether the synthetic chromosome is a MC or a DC basically impossible. This prevents one from using this technique to synthesize artificial metaphases to pre-train object detection models.

3.4 Training Faster R-CNN on a synthetic dataset

We trained Faster R-CNN (see Section 1.4.1 for a description of this model) on the synthetic datasets introduced in Section 3.1.2. To evaluate the relationship between model performance and the number of training image, we trained several models on 100, 200, 300, 400, 500, 1000 and 2000 images. Furthermore, we evaluated models trained on a specific dose on the test sets of other doses. For example, the 5Gy model was evaluated on the 1 Gy and 2 Gy test sets. This lets us see how much a change in dose affects model performance. This performance evaluation is summarized in Figure 3.9.

Because this is a simulation study intended to study the scaling behaviour of Faster R-CNN in rare object detection, we did not compute the counting distributions produced by our models. Instead, we computed a conventional object detection metric called Average Precision (AP), see Section 4.3.2 and[101] for detailed explanations on AP computation.

Overall, the conclusions are fairly intuitive: performance improves with the number of images in the training dataset. Cross-evaluations reveal that a model trained on a specific dose does not perform as well on other dose levels. Models always perform better in terms of AP at higher dose levels, where the proportion of DCs is higher.

We also note that training models at lower doses is difficult, because DCs become even more under-represented as the dose get lower. At 1 Gy, there might only be one DC every 20 metaphases, which means that the model does not get any gradient feedback for most images. We found that training models at doses lower than 1 Gy was unfeasible, as the trained model would not predict any bounding boxes. Another important detail is that

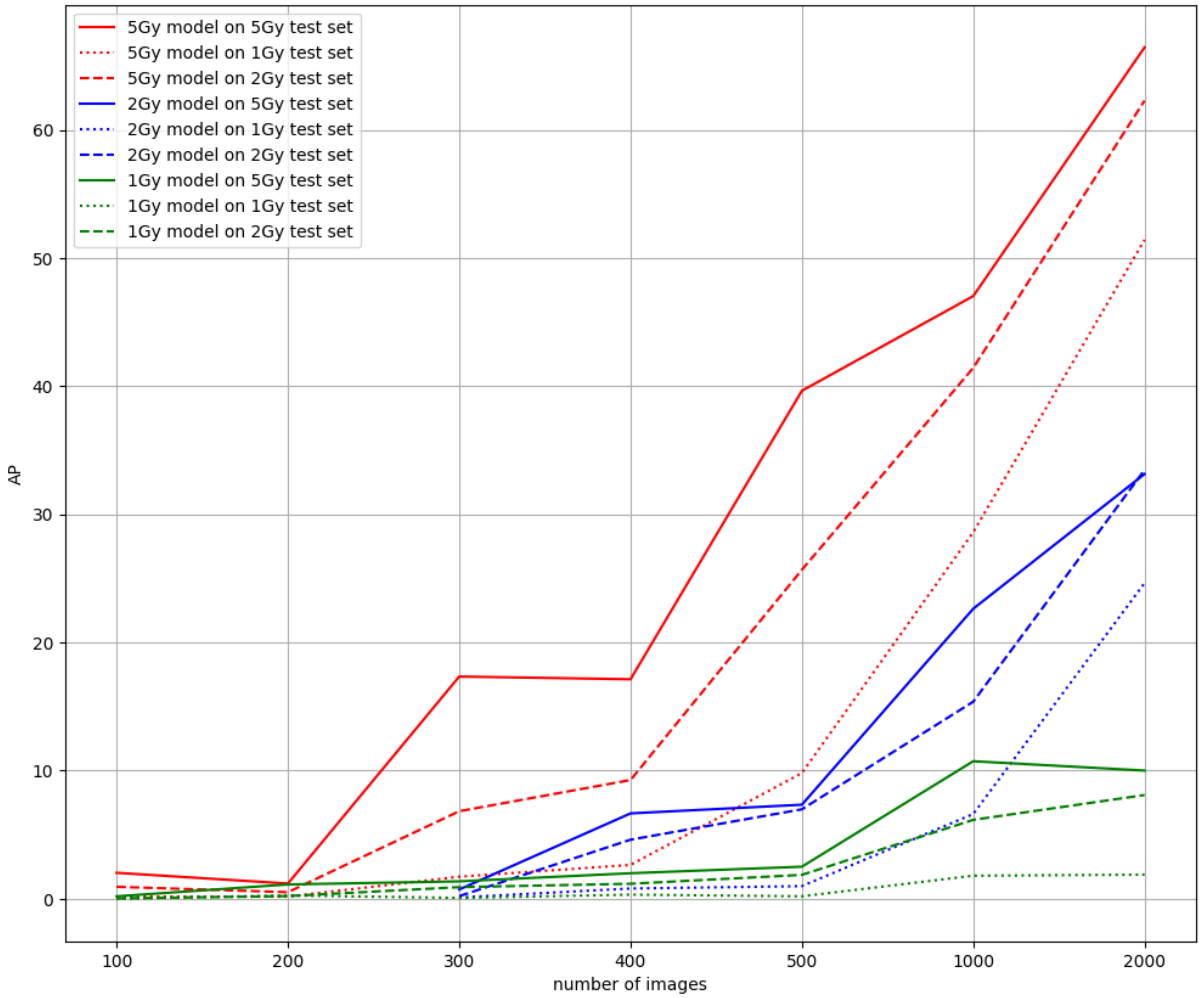


Figure 3.9 – Performance curves on simulated data, for models trained at 1, 2 and 5 Gy, with training dataset sizes ranging between 100 and 2000 images.x-axis is number of images, y-axis is Average Precision (AP), see Section 4.3.2 for an explanation of this object detection metric.

at low doses (less than 0.5 Gy), small datasets might not contain any aberration, making training completely impossible.

3.5 Conclusion

Training classifiers on datasets of chromosome patches provided limited result. Avoiding label leakage and building a truly representative datasets are very difficult tasks. This is further confirmed by the review presented in Chapter 2, which suggests that building

a two-stage ADS system requires a significant amount of engineering to deal with edge cases.

Training a DL-based object detection model to detect chromosomal aberrations seems much easier at first glance, as those can be trained end-to-end, and have been shown to reach a high level of performance in a wide variety of domains. However, a large labelled dataset is required. We used our classification dataset as a training dataset for image generation methods but the performance was unsatisfying. Both VAEs and GANs showcased low diversity, and low sample quality.

In the end, we used a "skeletonized" variant of this classification dataset to produce a supervised metaphase-level dataset. This was used to run a simulation study on the scaling performance of Faster R-CNN for DC detection. Faster R-CNN reached a non-trivial level of performance using as little as 1000 annotated images, without any architecture search or hyperparameter optimization. However, this result depends strongly on the dose of the dataset the model is trained on. Training on representative lower dose datasets performs worse, and requires a larger number of images. Furthermore, models trained on a specific dose perform differently on other doses, because of the difference in dicentric yields.

While our simulated data is much simpler than real world metaphases, this simulation study alleviated the largest concern, which was that DL-based object detectors would not be trainable with an attainable number of labelled images.

Chapter 4

End-to-end chromosomal aberration detection in metaphase images

As we saw in Chapters 2 and 3, designing a "pipeline" ADS system is difficult. The performance of every step is linked, sometimes in unexpected ways. ADS systems relying on conventional computer vision techniques may experience sudden performance degradation because of minute variations in image quality, as shown in the papers discussing DCScore performance in 2.6. To prevent this, ADCI relies on sophisticated image selection models, as explained in Section 2.4. On the other hand, DL-based models can be trained to be robust to changes in image quality. Our goal in this chapter is to use this ability to build simpler DC detection models.

We use a labelled object detection dataset to build a simpler, more effective model taking inspiration from keypoint regression. Our model predicts Gaussian "heatmaps" composed of circular spots localized in the center of the chromosomal aberration bounding box. Predicting one heatmap per class makes dealing with several aberration types easy. This model performs localization and classification in a single step and reaches a high level of performance, both in terms of object detection metrics and calibration curve fitting. We improve performance and provide a simple way to model prediction uncertainty by considering an *ensemble* of Unets.

The rest of this Chapter is organized as follows. First, we present some of our inspirations for our ensemble-based architecture. Then, we describe our training procedure, the evaluation of our model, and its object detection performance. We discuss the high requirements of biological dosimetry, and why simple object detection metrics are not necessarily informative in this context. Using domain knowledge, we improve the false positives performance of our ensemble, and present state-of-the-art results for fully-automated dose estimation. Finally, we provide visual explanations for some key abilities

of our model, namely its ability to reject non-chromosome objects, and the uncertainty modelling brought by ensembling.

4.1 Related works

4.1.1 Key-point regression in deep learning

In keypoint regression, Gaussian spots are predicted over specific landmarks of the image, like the eyes on a human face or the joints of a skeleton. It is a subtask of a large number of computer vision tasks like facial recognition or pose estimation [102], [103]. It can be solved with common image segmentation models by minimizing the L_2 distance between a predicted heatmap and the corresponding ground truth over a training dataset. While heatmap regression and object detection are different tasks at first glance, and heatmap regression does not predict bounding box extent, several authors have proposed a unified method that consists in predicting a center point and bounding box dimensions [67], or the corners of a bounding box as key-points [69] (see Section 1.4.2 for a detailed discussion). To our knowledge, key-point regression has never been used in biological dosimetry.

4.1.2 Object detection and counting

Object counting is a common computer vision task, and a wide variety of solutions have been suggested in the literature. Density-based methods aim to predict counts by integrating a density map [104], [105]. This class of methods is usually simple to implement and reaches a high level of performance. However, it does not detect objects, and summary statistics like Precision and Recall cannot be computed. Alternatively, counting can be solved as a subtask of object detection, by enumerating the bounding boxes belonging to a certain class. However, accurately predicting bounding boxes is more difficult than predicting a density map, and detection-based methods usually perform worse than density-based ones [106]. In [107], the authors propose a detection-based method that relies on predicting a Gaussian spot centered on the object, but does not predict bounding box extent. In [69], the author propose a model that uses the same heatmap-based technique, but also predicts a (height, width) tuple for a bounding box.

4.1.3 Model aggregation

A large number of papers have studied ensemble methods for neural networks, either to improve model calibration, for uncertainty modelling or to improve performance. The authors focus either on sampling a diverse set of models to build an ensemble, or on the properties of the aggregated prediction.

For classification models, Lakshminarayanan *et al.* [57] showed that ensembles of neural networks improved on the performance and calibration of single models. As an ensemble of M models requires M training runs, [58] showed that a carefully chosen learning rate schedule could encourage loss landscape exploration to get a collection of models with high diversity in a single training run by retrieving checkpoints. In [108], samples and checkpoints are re-weighted according to their performance, like AdaBoost [109].

In semantic segmentation, ensemble methods have received attention because they improve performance and provide a localized measure of uncertainty, usually by computing the entropy of the ensemble average for every pixel in the image. In [110], an ensemble of fully convolutional neural networks is used to segment aerial images. In [111], a diverse ensemble is built by training several Unets [10] with different encoders, and predictions are aggregated with a weighted average. In [112], the authors provide a review of ensemble methods for polyp segmentation.

For bounding-box based detection models, aggregation is required, as several boxes localizing the same object may overlap. Non-Maximum Suppression (NMS) [113] is then used and consists in sorting the boxes with respect to their confidence levels. Lower confidence boxes overlapping a high confidence box beyond a specific IoU (Jaccard index) threshold are discarded. In [114], box merging algorithms are explicitly considered in a particular model aggregation framework. The authors suggest computing an "average" box by weighting coordinates based on the box confidence.

Approximate Bayesian deep learning

Common techniques used to sample the posterior distribution are intractable for modern neural networks given their large parameter counts. In [7], Mandt *et al.* demonstrate that SGD can be seen as an Orstein-Uhlenbeck process with some limit Gaussian distribution. SGD can be seen as Langevin sampling of the posterior weight distribution, and SGD samples can be used to estimate the mean and covariance of the Gaussian posterior distribution. This provides a cheap and simple way to retrieve samples of this distribution

during the training procedure.

in [59], Garipov *et al.* al show that local minima are connected by low-loss paths, and that one could average weight vectors along deterministic trajectories between those local minima to improve performance. In [13], Izmailov *et al.* confirm the theoretical analysis of [7] by showing that averaging SGD iterates leads to wider minima and improved generalization in practice.

Using the theoretical insights explained in [7], Maddox *et al.* [6] approximate the limit posterior weight distribution with a Gaussian distribution, where the covariance matrix is defined as the covariance of the last gradient descent iterates. New sets of weights can be sampled from this posterior distribution for ensemble and uncertainty estimation. In [115], the authors go further by sampling an ensemble from several modes of the posterior weight distribution to increase ensemble diversity and therefore performance.

Finally, other approximations of the posterior weight distribution have been proposed. In [116], the authors use Kalman filtering to derive a sequential estimate of the posterior distribution over the weights.

4.2 Methods

In this section, we first give a precise definition of the data and model, including the loss for training. Second, we explain our aggregation procedure, going from a set of continuous heatmap predictions for a single image to a set of binary decisions maps and finally, to an aggregated ensemble-level prediction. Finally, we explain the PCA-based visualization techniques used to justify the performance gains of the ensemble and its robustness to the presence of debris in metaphase images.

4.2.1 Keypoint regression with heatmap regression models

In Heatmap Regression Models (HRMs), objects of interest are represented as Gaussian spots. The model is trained to predict spot positions in the image domain, with a labelled dataset $\mathcal{D} = \{(x_1, y_1), \cdots, (x_n, y_n)\}$ comprised of n realizations of a pair of random variables (X, Y). For each image x_i, we have $x_i(u, v) \in [0, 1]$ at each location $(u, v) \in \Omega$, where Ω denotes the image grid of size $|\Omega| = H \times W$.

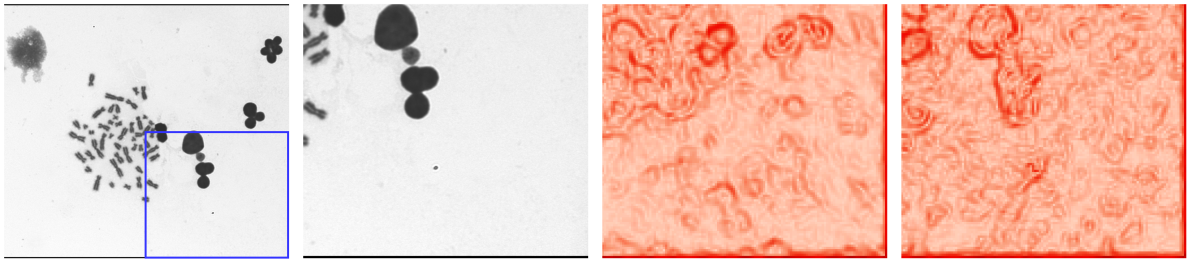


Figure 4.1 – Visual comparison between regularized and unregularized model. First image from the right: input image, second: bottom right crop, third: gradient norm of the prediction for the unregularized model, fourth: gradient norm of the prediction for the regularized model.

Model design and sparsity promoting loss function

Our heatmap regression model is a convolutional neural network $\phi_\theta(x) : x \in [0,1]^{H\times W} \rightarrow y \in [0,1]^{H\times W}$. The final output is constrained between 0 and 1 with a sigmoid layer. We use the Unet architecture [10] to predict a low resolution heatmap. The image y_L is of size $(H/L, W/L)$ for some arbitrary downsampling factor L, as the location accuracy provided by the highest resolution output is not useful. For Unet-based architectures, images are usually downsampled (or upsampled, in the decoder) by a factor of 2 at each layer: at layer l of the encoder, features have a spatial size of $H/2^l \times W/2^l$. For the sake of simplicity, notations are given in the single-channel case. Additional classes of aberrations (like fragments) are modeled with additional channels, so that a third index is added. In the remainder of this Chapter, we consider two aberration classes, DCs and fragments. The parameters θ are learned by solving the following optimization problem:

$$\theta^\star = \arg\min_\theta \sum_{i=1}^{n} \mathcal{L}(\phi_\theta(x_i), y_i) + \lambda\, \mathcal{R}(\phi_\theta(x_i)), \tag{4.1}$$

where λ is an hyperparameter that balances the data fidelity term and the regularization term. In our modelling approach, the data fidelity term $\mathcal{L}$ has the following form:

$$\mathcal{L}(\phi_\theta(x_i), y_i) = \|\phi_\theta(x_i) - y_i\|_2^2 \tag{4.2}$$

Because the number of aberrations is very low compared to the number of pixels in the image, the background is expected to be 0, except in a small number of "hot" spots corresponding to locations containing aberrations. The Sparse Variation (SV) regularizer

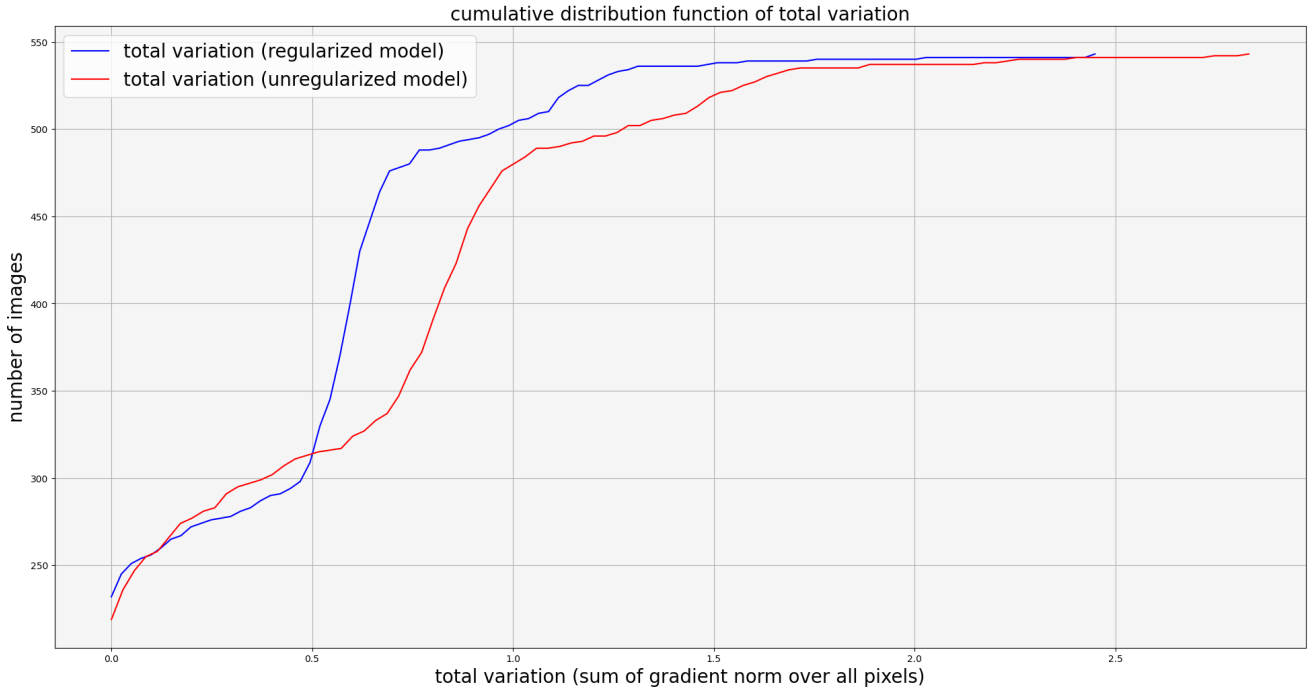


Figure 4.2 – For each image, we compute the total variation of its prediction for the regularized and unregularized model, as shown in Figure 4.1. This figure represents the Cumulative Distribution Function (cdf) of the total variation over all images in the test set.

(4.3) has been specifically considered here to encourage the emergence of a very small number of "hot" spots as aberrations are rare events in Giemsa-stained images. This regularizer is defined as [11]:

$$\mathcal{R}(\phi_\theta(x_i)) = \sum_{(u,v)\in\Omega} \sqrt{\rho^2 \|\nabla_{u,v}\phi_\theta(x_i)\|_2^2 + (1-\rho)^2 \phi_\theta(x_i)^2(u,v)}, \tag{4.3}$$

where ρ is a parameter that balances the sparsity and the smoothness terms in the predicted heatmap. The components of the gradient vector are computed with respect to the image coordinate axes as follows:

$$\nabla_{u,v}\phi_\theta(x_i) = \begin{bmatrix} \phi_\theta(x_i)(u,v) - \phi_\theta(x_i)(u+1,v) \\ \phi_\theta(x_i)(u,v) - \phi_\theta(x_i)(u,v+1) \end{bmatrix}. \tag{4.4}$$

As this is a simple linear transformation of the image, computational overhead is minimal.

The criterion (4.2) is highly non-convex because of non-linearities in ϕ_θ. Therefore, finding a global minimum is hopeless. Nevertheless, a good local minima may be found

using iterative first order methods, usually some variant of SGD. The exact gradient of the training criterion with respect to θ is estimated on a random subset $\mathcal{J}$ of the complete dataset $\mathcal{D}$, because of memory constraints:

$$\nabla_\theta \sum_{i=1}^{n} \mathcal{L}(\phi_\theta(x_i), y_i) + \mathcal{R}(\phi_\theta(x_i)) \simeq \nabla_\theta \sum_{j=1}^{|\mathcal{J}|} \mathcal{L}(\phi_\theta(x_j), y_j) + \mathcal{R}(\phi_\theta(x_j)). \tag{4.5}$$

In Figure 4.2, we reported two curves corresponding the cumulative distributions functions of the total variation images computed over the prediction maps regularized with the sparse variation regularizer (blue curve) and without (red curve). See 4.1 for an example of images on which this total variation is computed.

4.2.2 Implicit ensembling of neural networks

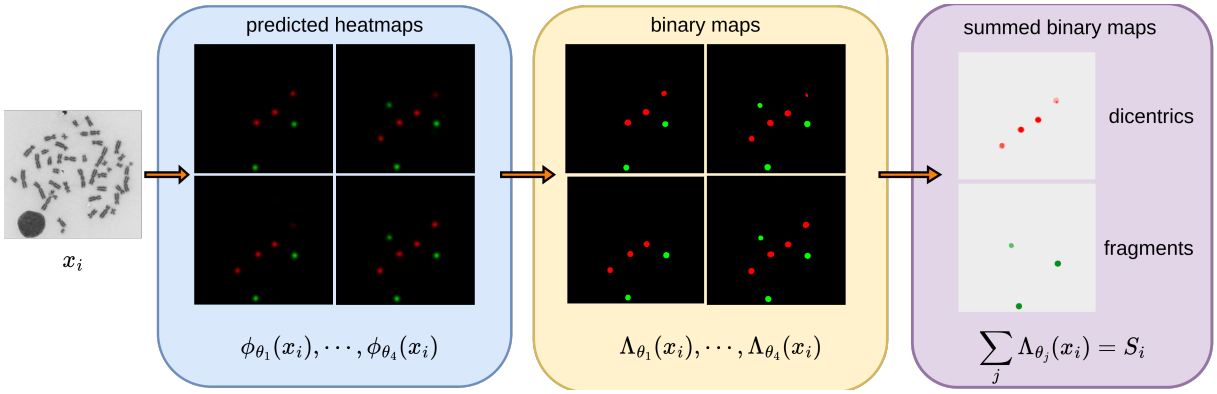


Figure 4.3 – Vote-based aggregation of checkpoints. DCs predictions are plotted in red, and fragment predictions are plotted in green. For every image x_i, the heatmap prediction $\phi_\theta(x_i)$ is binarized (giving $\mathbf{\Lambda}_\theta(x_i)$) with a confidence threshold T_C. Those maps are summed (giving S_i), and regions of the image receiving more than T_A votes are considered as detections. Darker shades of red and green indicates region of the images receiving more votes.

To reduce the number of false positives (i.e, improve Precision) at a fixed recall level, we investigated the ensemble method introduced in [57]. Because training a deep neural network is a stochastic process, successive training runs of the same model tend to explore different regions of the parameter space. Those local minimas are usually very close in terms of validation loss, but their predictions are not identical. This behavior has received significant attention in the literature [58], [59]. As shown in [58], it may not even be necessary to run several successive training runs. A carefully chosen learning rate schedule

may be enough to achieve enough parameter space exploration to build a diverse ensemble from checkpoints of a single training run. In our case, we even find that the gradient noise introduced by stochastic batch sampling leads to sufficient checkpoint diversity for agregation to be worthwile without any specific learning rate schedule. Therefore, we do not have to deal with the training instability mentioned in [6].

More formally, SGD can be seen as a Langevin sampling of the posterior weight distribution $p(\theta|\ \mathcal{D})$ [7]. To avoid excessive autocorrelation between samples, we store the vectors θ at the end of each epoch, instead of every gradient step. This can be interpreted as a variant of *thinning*, also used in Monte Carlo Markov Chain inference. For each image x_i, $i \in \{1, \cdots, n\}$ in the test set, we consider a set $\{\phi_{\theta_1}(x_i), \cdots, \phi_{\theta_M}(x_i)\}$ of predictions, where M is the number of predictions (or models). Using a confidence threshold T_C, we build a set $\{\mathbf{\Lambda}_{\theta_1}(x_i), \cdots, \mathbf{\Lambda}_{\theta_M}(x_i)\}$ of M different binary predictions for each test image x_i, that we sum over all members of the ensemble at each location $(u, v) \in \Omega$ as follows:

$$\boldsymbol{S}_i(u, v) = \sum_{m=1}^{M} \mathbf{\Lambda}_{\theta_m}(x_i)(u, v),\ (u, v) \in \Omega\,. \tag{4.6}$$

Finally, we set an agreement threshold T_A to compute the agreement between the artificial "experts". The setting of thresholds T_A and T_C impact the final decision. If the confidence threshold T_C is high and the voting threshold T_A is low, the decision will be made from a small set of experts. Otherwise, a small value of T_C but a high voting threshold T_A means that low confidence predictions are considered, but a higher agreement between them is needed to confirm a detection. In the end, we get a precision surface depending on T_C and T_A. Our aggregated decision for any image x_i is a binary image $\boldsymbol{D}_i$ such that value at location (u, v) is 0 if no aberration is predicted, and 1 otherwise (See Figure 4.3 for illustration):

$$\boldsymbol{D}_i(u, v) = \mathbb{1}[\boldsymbol{S}_i(u, v) > T_A],\ (u, v) \in \Omega. \tag{4.7}$$

The agreement threshold T_A can be adjusted by the end user to optimize either Precision or Recall scores, like the confidence threshold T_C. We discuss the effects of choosing a specific threshold in Section 4.3.2.

4.2.3 Setting of model parameters

We trained Unet for $N_e = 100$ epochs epochs with Adam [12], with a constant learning rate of 3×10^{-4}, a weight decay parameter of 0.1 and a batch size of 12 on a single

Tesla V100. The learning rate was unchanged during training to ensure parameter space exploration, using an analoguous reasoning to the one provided in [13].

We did not use data augmentation for two reasons. First, we found that the wide variety of chromosome morphology and orientations in our dataset was enough for our model to learn this invariance. We did not observe detection failures based on object orientation. Second, more agressive data augmentation like noise or blurring quickly made monocentrics and dicentrics indistinguishable. Training stability was very sensitive to the variance of the blurring kernel or the Gaussian noise, because accurate chromosome classification relies on very small details.

We predict a lower resolution heatmap of size $H' = 224, W' = 252$, where height and width are downsampled by a factor of 4. While batch sampling (and therefore parameter space exploration) is randomized, parameter initialization is fixed between training runs. We ran a grid search with $\log_{10}$ spacing for λ regularization parameter with 10 and 10^{-4} as upper and lower bound of the search interval. Training was implemented in PyTorch [20], and uses `segmentation_models_pytorch` implementation of Unet.

4.2.4 Visualization of the training dynamics of single model

As an additional visual explanation for aggregation performance gain, it may be informative to display training trajectories in feature space. Classification networks output a single classification vector per image, so that plotting training dynamics over time using dimensionality reduction is relatively easy (see [117]). A scatterplot of classification vectors embedded in a lower dimension at each epoch provides a good view of how classes are progressively separated during training. Usually, UMAP [118] is used, which requires a pairwise distance matrix between classification vectors. However, this visualization does not work for models outputting a probability distribution for all locations u, v of the input image, like Unet.

To address this issue, we adapt the approach [117] to our context. We consider feature maps as bags of independent feature vectors. Figure 4.4 provides a visual summary of our approach. We do not retain feature vectors for all locations u, v in the feature map. Instead, we only select feature vectors corresponding to the locations of aberrations, and retrieve some feature vectors at random 'background' (i.e., where there are no aberrations) locations. This bag of features is projected on the 2D plane using its PCA decomposition. By retrieving the same locations across several training steps, we can visualize how the both aberration classes and the background are separated during training. An SVM clas-

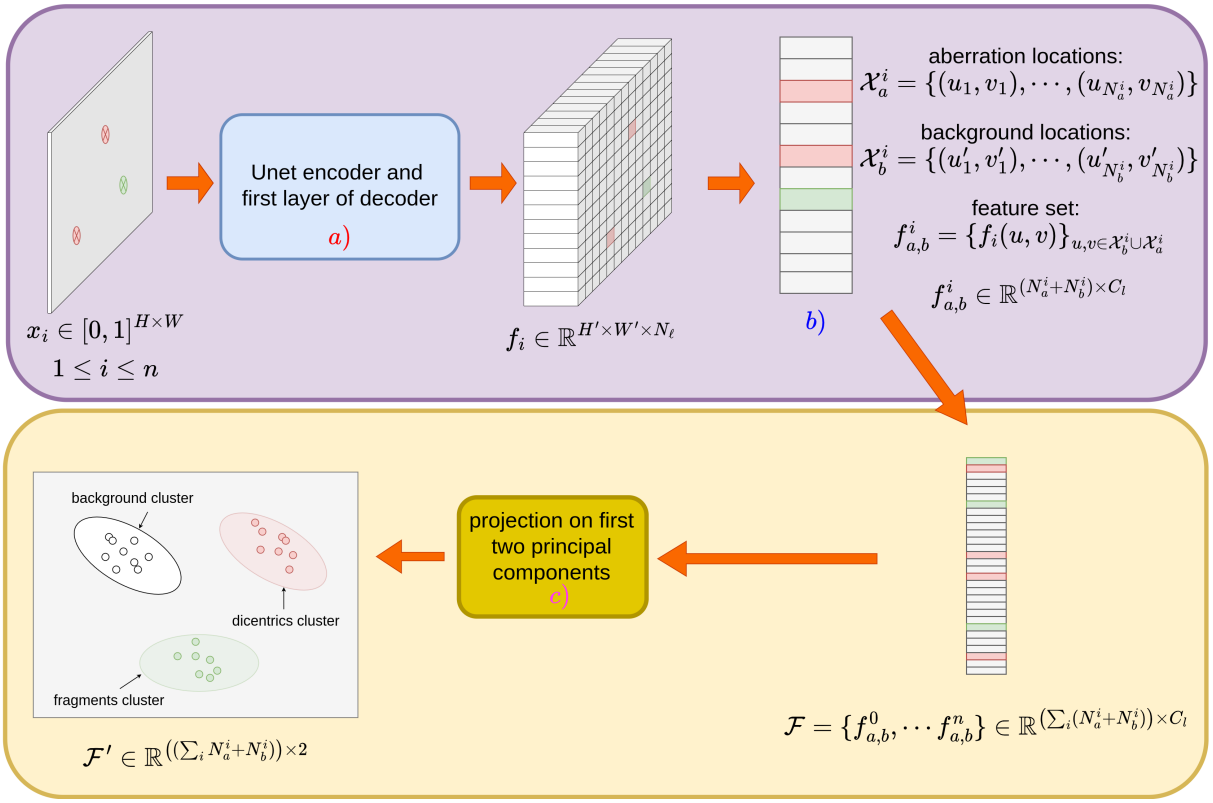


Figure 4.4 – Procedure used to display feature separation in the latent space of the last decoder block for a single epoch (i.e a single weight vector θ. For all images $x_1, \cdots, x_n$, the feature maps produced by the last layer of the decoder are retrieved and treated as a set of independent, C_l-dimensional feature vectors. Using PCA dimension reduction, we produce a 2D scatterplot that shows how the model separates the different classes (background, dicentrics, fragments) across training epochs. Note that the eigenvectors used for this dimensionality reduction are computed over *all* epochs of training.

sifier fitted on the embeddings retrieved for a single epoch is used to map regions of the latent space to a specific class, which helps visualize the dynamics of training. The rest of this section gives a formal overview of our visualization technique.

Formally, for an input image x_i of size $H \times W$, the ℓ-th layer of our Unet produces a feature volume f_i^ℓ of size $H' \times W' \times N_\ell$ (see Section 4.4.3 and Figure 4.4 (a)). Here, we choose the second-to-last layer of the decoder and we drop the superscript ℓ to improve readability, so that $f_i^\ell = f_i$. For the last layer, $N_\ell = 128$ (see illustration in Figure 4.4). We retrieve the set of feature vectors that correspond to the spatial locations of aberrations (dicentrics and fragments) in image x_i. For a set of N_a^i aberrations located at $\mathcal{X}_a^i = \{(u_1, v_1), \cdots, (u_{N_a^i}, v_{N_a^i})\}$ in image x_i, we build a set of feature vectors

$f_a^i = \{f_i(u_1, v_1), \cdots, f_i(u_{N_a^i}, v_{N_a^i})\} \in \mathbb{R}^{N_a^i \times N_\ell}$. We retrieve the feature vectors corresponding to all aberrations in every image the test set. We also sample an additional set of N_b^i background pixels, denoted as f_b^i at locations $\mathcal{X}_b^i = \{(u'_1, v'_1), \cdots, (u'_{N_b^i}, v'_{N_b^i})\}$. Those locations are randomly sampled, provided the locations do not correspond to aberration pixels. Therefore, they correspond to background, monocentric or debris. Finally, we define $f_{a,b}^i = f_a^i \cup f_b^i$ so that the total number of feature vectors in f_i is $N_a^i + N_b^i$, as shown in Figure 4.4b). It is worth noting that $f_{a,b}^i$ is a subset of the complete feature map f_i. This makes the visualizations described less cluttered, and reduces computation time. Finally, this feature set $f_{a,b}^i$ is retrieved for each image x_i in the test set, to build a large feature set $\mathcal{F}_e = \{f_{a,b}^0 \cdots f_{a,b}^n\}$, where e corresponds to the set of model parameters retrieved at epoch e.

The set of feature vector $\mathcal{F}_e$ is retrieved for each epoch $e \in \{1, \cdots, N_e\}$. These sets are concatenated in global set $\mathcal{F} = \{\mathcal{F}_1, \cdots, \mathcal{F}_{N_e}\}$. Although all the subsets of $\mathcal{F}$ correspond to the same locations, they are different at each epoch e because of the stochasticity of gradient descent. PCA can be used to visualize those feature sets as 2D scatterplots, and in particular to check how well aberrations are separated from the background at every epoch. The set $\mathcal{F}_1$ is chosen as a reference feature set and $\mathcal{F}_e$, $e \in \{2, \cdots, N_e\}$ is registered with respect to $\mathcal{F}_1$ using the Procrustes method [119]. This guarantees that latent space scale shifts or rotations are removed for visualization. A PCA decomposition is computed on $\mathcal{F}$, and for each epoch e, each feature set in $\mathcal{F}_e$ is projected on the first two principal components of this decomposition. This provides a 2D visualization of the trajectory of each feature vectors during training.

Furthermore, we train a kernel SVM classifier p_e on the 2D embeddings of $\mathcal{F}_e$ to predict which aberration class corresponds to a location in 2D embedding space. This classifier takes a 2D embedding (a vector of $\mathcal{F}_e$) as an input, and outputs a probability distribution over three classes: background, DCs and fragments. For all epochs $1 \leq e \leq N_e$, we train a different classifier, and predict a probability distribution over a grid that samples the 2D aberration space uniformly. For all positions (u, v) of this grid, a probability distribution over the total number N_r aberration classes $p_e(u, v, r) \in [0, 1]$, $r \in \{0, \cdots, N_r\}$, $\sum_{r=1}^{N_r} p_e(u, v, r) = 1$ is predicted at epoch e. As all the point clouds are aligned and a single set of principal components is computed for all time steps, the changes in the decision boundary from one epoch to the next can be solely attributed to the dynamics of training.

Finally, to visualize the displacement of class boundaries across training, we define the

averaged classifier:

$$\bar{p}(u,v) = \frac{1}{N_e}\sum_{e=1}^{N_e} p_e(u,v). \tag{4.8}$$

Visualizing the spread of the distribution of $\bar{p}$ can be done by computing the entropy of the distribution predicted by the averaged classifier:

$$H(\bar{p})(u,v) = -\sum_{r=1}^{N_r} \bar{p}(u,v,r)\log\bar{p}(u,v,r). \tag{4.9}$$

4.3 Materials

4.3.1 Data description

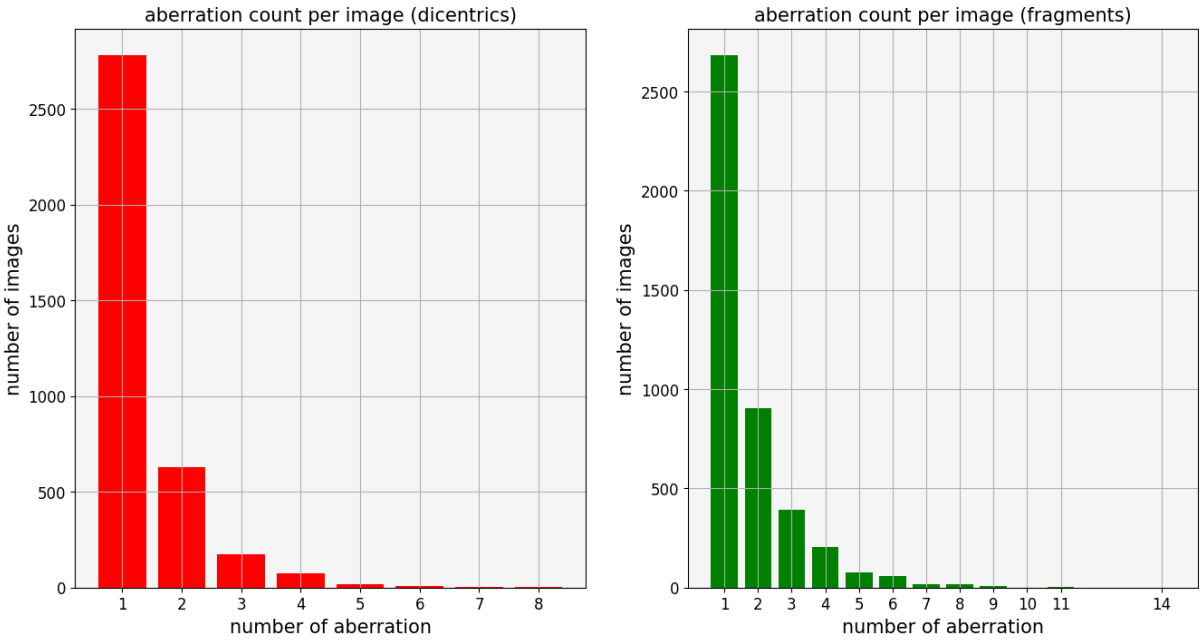


Figure 4.5 – Repartition of images into aberration counts bins.

Our training dataset is composed of 5430 labelled images of size $H = 888, W = 1008$, padded to $H = 896, W = 1008$ to ensure that downscaling has an integer height and width. Labels are binary images with size $H' = 202, W' = 252$, taking value 0 everywhere except at the center of chromosomal aberrations (roughly between the two centromeres for a DC), where it takes value 1. There is one binary image per aberration classes for each image, so that aberration classification is possible. Chromosomal aberrations are

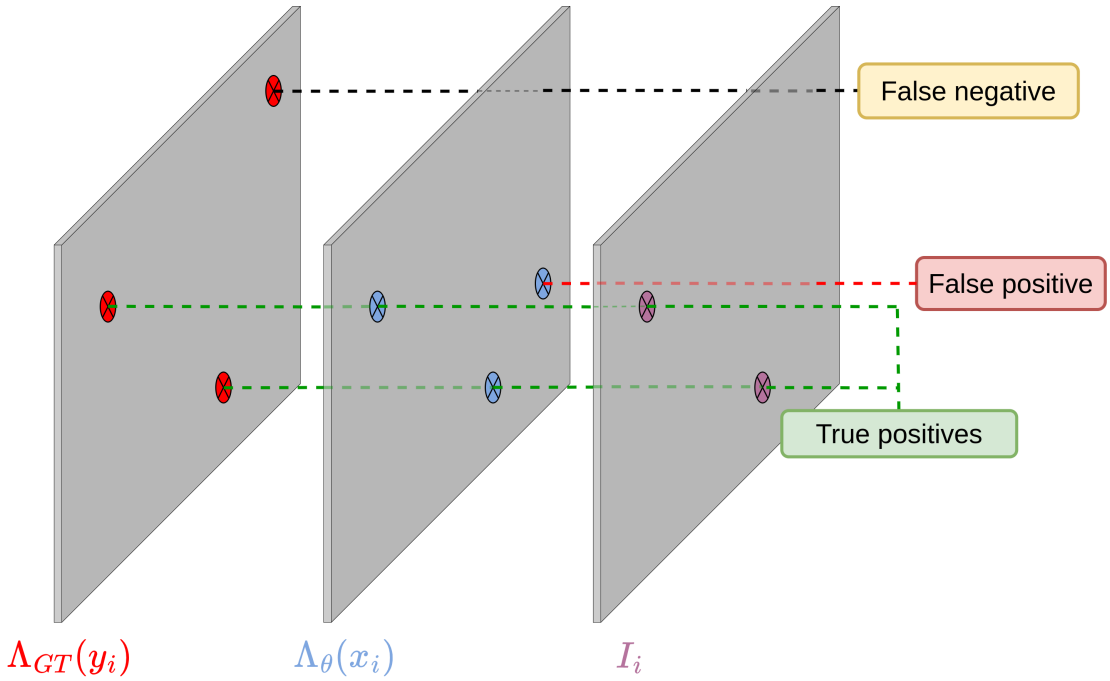


Figure 4.6 – Sketch of model evaluation. The intersection $\boldsymbol{I}_i$ between the binarized ground truth $\boldsymbol{\Lambda}_{GT}(y_i)$ and the binarized prediction map $\boldsymbol{\Lambda}_\theta(x_i)$ is computed. Objects appearing in both are true positives, objects appearing only in $\boldsymbol{\Lambda}_\theta(x_i)$ are false positives, objects appearing only in $\boldsymbol{\Lambda}_{GT}(y_i)$ are false negatives. In this case, we have two true positives, 1 false negative and 1 false positive, so that Precision is $TP/(TP+TP) = 2/3$ and Recall is $TP/(TP+FN) = 2/3$

the only labelled objects, neither debris nor monocentric chromosomes are labelled. We chose this labelling scheme instead of semantic segmentation or bounding boxes as it lead to the lowest labelling time, which in turn meant a greater number of images could be labelled for the same labelling budget. For the same reason, we did not label debris or monocentric chromosomes. This also prevented the discovery of trivial models where chromosomes would be detected but always labelled as monocentric, as they outnumber dicentric ones by an extremely large margin. As explained in Section 4.2.1, this binary image is blurred with a Gaussian kernel, to reduce the underrepresentation of the labels against the background.

Images have been selected so that each images contains only the chromosomes corresponding to a single cell, i.e., no image contains more (or less) than 46 chromosomes. Our dataset contains 5021 dicentrics and 7540 fragments, Figure 4.5 shows the repartition of images of images into aberration counts bins. Images with a high aberration count are

much rarer than images with a low aberration count. This training dataset is a subset of a much larger archive of patient data (containing around 80k images). It contains all images with at least one aberration in the archive. This reduces training time considerably, and prevents the discovery of trivial models where no object is ever predicted. However, this also means that it is not an accurate representation of real-world metaphase. As a normal metaphase contains 23 chromosome pairs, this means that even in this case, the overwhelming majority of chromosomes are healthy (i.e monocentric) ones. In our evaluation setting, the training set consists of 80% of those image. 10% of the images are retained for a validation set, used for hyperparameter selection. Finally, 10% of the data is held as a test set for a fair performance evaluation.

As this training dataset is not representative of a real-world exposition, we use another dataset to estimate the calibration curve of our model. This dataset was built by collecting metaphases from samples irradiated at specific, known doses. The aberrations in this dataset are not labelled. It contains 21215 metaphases taken from samples irradiated at 0 Gy, 0.1 Gy, 0.2 Gy, 0.3 Gy, 0.5 Gy, 0.7 Gy, 0.9 Gy, 1 Gy, 1.5 Gy, 2 Gy, 3 Gy and 4 Gy.

4.3.2 Evaluation metrics

Performance of a single model

While our predicted heatmap $\phi_\theta(x_i)$ can take any value between 0 and 1 at each spatial position (u, v) in the image, ultimately a binary decision needs to be taken with regard to the presence or absence of aberration at location $(u, v) \in \Omega$. In the next step $\phi_\theta(x_i)$ is used to build a binary map $\mathbf{\Lambda}_\theta(x_i)$ given an arbitrary threshold T_C:

$$\mathbf{\Lambda}_\theta(x_i)(u, v) = \mathbb{1}[\phi_\theta(x_i)(u, v) > T_C],\ (u, v) \in \Omega,\ x_i \in \{x_1, \cdots, x_n\}, \tag{4.10}$$

where $\mathbb{1}[.]$ is the indicator function. Furthermore, a ground truth binary map is defined given a confidence threshold T_{GT}, which is set to a small fixed value (e.g., 0.01).

$$\mathbf{\Lambda}_{GT}(y_i)(u, v) = \mathbb{1}[y_i(u, v) > T_{GT}],\ (u, v) \in \Omega,\ \ y_i \in \{y_1, \cdots, y_n\}\,. \tag{4.11}$$

Once a binary decision for the presence of an object is made at each location in the image, we define true positives, false positives and false negatives. In object detection, true positives are usually defined up to a small location error, as matching the ground truth perfectly would be too stringent. In our case, the spots are small compared to the

object size so that the position error remains very small even in the cases where the intersection between the predicted and ground truth spot is the smallest possible one (one pixel) (see Figure 4.6). Therefore, we consider any overlap between a prediction and a ground truth spot to be a true positive, as long as this ground truth spot has not been predicted before. If this is the case, the prediction is considered to be a false positive. Predicted objects that do not overlap ground truth spots are also considered to be false positives. Finally, objects in the ground truth heatmap that are not predicted by the model are considered to be false negatives. True negatives are ill-defined in object detection, and are not considered. With those three values, we can compute Precision and Recall.

More formally, the intersection image $\boldsymbol{I}_i$ between $\boldsymbol{\Lambda}_\theta(x_i)$ and $\boldsymbol{\Lambda}_{GT}(y_i)$ is computed as the pixel-wise product of ground truth and prediction

$$\boldsymbol{I}_i(u,v) = \boldsymbol{\Lambda}_\theta(x_i)(u,v) \times \boldsymbol{\Lambda}_{GT}(y_i)(u,v),\ (u,v) \in \Omega,\ \ i \in \{1, \cdots, n\}\,. \tag{4.12}$$

Finally, we compute the number of connected components $N_{cc}(\boldsymbol{I}_i)$, $N_{cc}(\boldsymbol{\Lambda}_\theta(x_i))$ and $N_{cc}(\boldsymbol{\Lambda}_{GT}(y_i))$ in $\boldsymbol{I}_i$, $\boldsymbol{\Lambda}_\theta(x_i)$ and $\boldsymbol{\Lambda}_{GT}(y_i)$, respectively. The number of true positives, false positives and false negatives are defined as follows:

$$\begin{cases} TP = N_{cc}(\boldsymbol{I}_i)\,, \\ FP = \max(0, N_{cc}(\boldsymbol{\Lambda}_\theta(x_i)) - TP)\,, \\ FN = \max(0, N_{cc}(\boldsymbol{\Lambda}_{GT}(y_i)) - TP). \end{cases} \tag{4.13}$$

Hence, we compute Precision $= \frac{TP}{TP+FP}$ and Recall $= \frac{TP}{TP+FN}$ for a single $(\boldsymbol{\Lambda}_\theta(x_i), y_i)$ pair. Note that Precision and Recall are functions of the chosen confidence level T_C; a higher confidence threshold increases Precision but decreases Recall. In what follows, we report results for a set of confidence thresholds $\{T_{C_1}, \cdots, T_{C_{10}}\}$ to analyze this tradeoff. Moreover, we decided to report the Precision and Recall scores separately instead of providing an aggregated metric like Average Precision (AP), as the tradeoff between those metrics in the context of biological dosimetry is especially important.

Because training is a stochastic process, the maximum predicted probability over the test set is not exactly 1; different models may get a different maximum confidence value. In other words, two sets of weights θ and θ' will give two different maximum probabilities p_θ and $p_{\theta'}$, so that for each model, the performance metrics are computed over different confidence thresholds. Therefore, each performance metric curve is linearly interpolated

over a common confidence grid $\{0.1, 0.2, \cdots, 0.9\}$. For the set of confidence thresholds that exceed the maximum probability over the complete test set, Precision is not defined. In this case, it is arbitrarily chosen to be 1 (and Recall is 0). To provide a metric showcasing performance variation across training, we reported performance quantiles (5% and 95%) for each threshold. This means that, for each threshold, 50 Precision and Recall values are computed (one for every considered epoch) and the aforementioned quantiles of those values are reported.

Performance of model ensemble

For the ensemble, we use the same evaluation procedure as in the single model case (described in Section 4.3.2), but with the agregated binary decision map $\boldsymbol{D}_i$ described in Section 4.2.2. However, as explained in Section 4.2.2, the performance of the ensemble depends on an agreement threshold T_A and a confidence threshold T_C. Therefore, instead of a Precision curve, we get a Precision surface, which makes comparison with the single model case more difficult. Instead, we set a specific vote threshold, and report the same Precision curve as in the single model case. Finally, to evaluate the sensitivity of the performance to the sampling of the ensemble, we sampled 100 random ensembles, and computed q_{05} and q_{95} for every confidence threshold in the grid mentioned in the Section 4.3.2.

4.4 Experimental results

In this section, we discuss model performance, both in term of object detection and calibration curve estimation. First, we discuss the results of the single-model training and the impact of regularization term. We also show a visualization which suggests that our model is robust to the presence of debris, without the need for specific labelling or architectural choices. Finally, we discuss the performance improvements obtained with the ensemble approach, and we provide PCA-based approach to visualize model feature trajectories during training.

4.4.1 Performance of single model

In Table 4.2, the single non-regularized model already achieves significant gain in terms of Precision and Recall when compared to Metafer in the case of MC versus DC classifica-

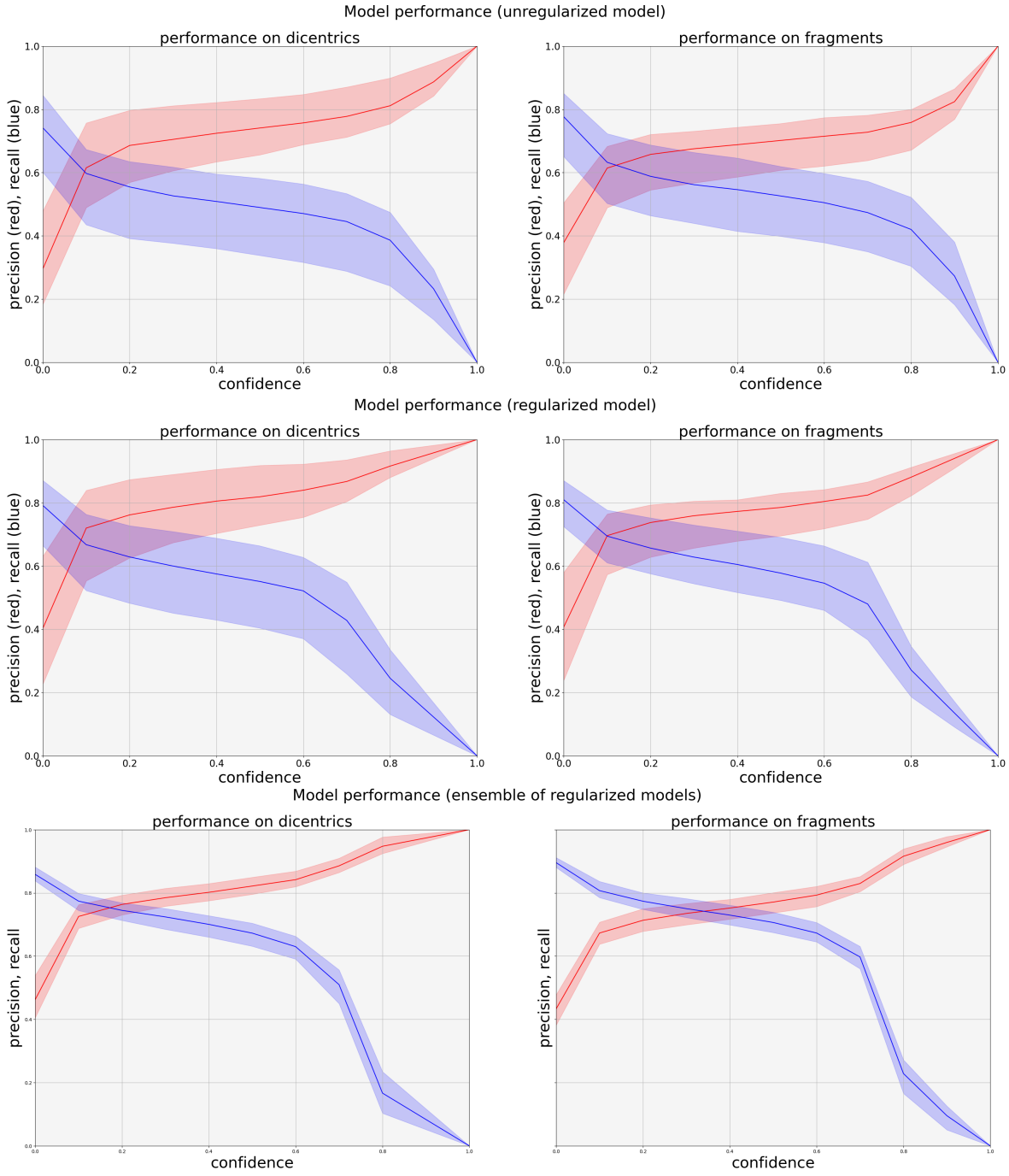


Figure 4.7 – Precision, Recall and False Discovery Rate (FDR) as functions of confidence for DCs (left) and fragments (right). Top: performance summary for the unregularized model (i.e $\lambda = 0$ for the sparse variation term). Middle: performance summary for $\lambda = 0.2, \rho = 0.1$. Bottom: performance summary for the ensemble of checkpoints from the training of the regularized model for a threshold of 2 votes. Shaded area indicates the $[q_{0.05}, q_{0.95}]$ inter-quantile interval, computed respectively over the last 50 checkpoints for single models, and over a 100 random samples of 10 checkpoints for the bottom plot (ensemble).

tion (Metafer is not designed to detect fragments), although some performance variation is noticeable during training, as confirmed by the inter-quantile range of performance. This variation suggests that there is sufficient parameter space exploration to get enough prediction diversity for aggregation to be worthwhile.

Metafer relies on conventional computer vision techniques. The chromosome objects in the metaphase are segmented and then classified. Segmentation errors can lead to classification errors, for example when one chromosome is split during segmentation. Overall, the performance of both tasks (segmentation and classification) is not very robust to variations in image quality induced by variations in acquisition circumstances like illumation or staining quality. For example, for the segmentation task, the Recall of Metafer lies between 35% and 75%. For the classification task, The Recall is around 35%, and the Precision around 40%. Note that the performance figures given for Metafer in Table 4.2 are not computed on the dataset mentioned in Section 4.3.1, but taken from [93] instead. Although the performance of Metafer is uncertain, and we did not compute it on our dataset, we feel confident that our model brings a very significant improvement in chromosomal aberration, as the uncertainty around the ensemble performance is low enough that even in the worst case scenario, it achieves very significant performance improvements over Metafer.

Finally, training with a loss that promotes sparsity improves Precision and Recall for fragments and DCs. this improvement in precision is especially relevant for automated use, as keeping the number of false positives low is required to avoid overloading care facilities. Because all chromosomes may not be correctly retrieved (Recall is less than 1), this model tends to underestimate doses.

	M1 (unregularized)	M2 (regularized)	ensemble (3 votes, $T_C = 0.5$)
Pr (frags)	70.8% (62.0,77.4)	79.3% (71.8, 84.2)	81.0% (78.5, 83.3)
Rc (frags)	49.4% (37.8, 59.7)	55.0 % (46.0, 66.3)	65.5 % (62.4, 68.5)

Table 4.1 – Comparison between model 1 (denoted as M1, $T_C = 0.6, \lambda = 0$), model 2 (denoted as M2, $T_C = 0.6, \lambda = 0.2, \rho = 0.1$) and ensemble (4 votes, $T_C = 0.5$) for the fragment class. $[q_{5\%}, q_{95\%}]$ interval is reported in parenthesis. For model 1 and model 2, this interval is computed over the last 50 epochs of training. For the ensemble, it is computed over a 100 randomly sampled ensembles (ensembles are sampled from checkpoints during training). All performance metrics are computed on the test set.

	M1 (unregularized)	M2 (regularized)	ensemble (3 votes, $T_C = 0.5$)	Metafer
Pr (dics)	76.6% (68.8, 84.7)	83.6% (75.4, 92.2)	**85.8%** (83.7, 88.3)	$\sim 40\%$
Rc (dics)	45.2% (31.6, 56.4)	51.2% (37.0, 62.7)	**61.7%** (57.1, 65.8)	$\sim 35\%$

Table 4.2 – Comparison between model 1 (denoted as M1, $T_C = 0.6, \lambda = 0$), model 2 (denoted as M2, $T_c = 0.6, \lambda = 0.2, \rho = 0.1$), **ensemble** (4 votes, $T_C = 0.5$), and Metafer for the DC class (where Metafer performance is available). All performance metrics are computed on a separate test set. Metafer performance is retrieved from previous work, and was not evaluated on the dataset used in this Chapter. Metafer performance should only be taken as a rough point of reference, see Section 4.4.1 for additional discussion of this point.

4.4.2 Performance of model ensemble

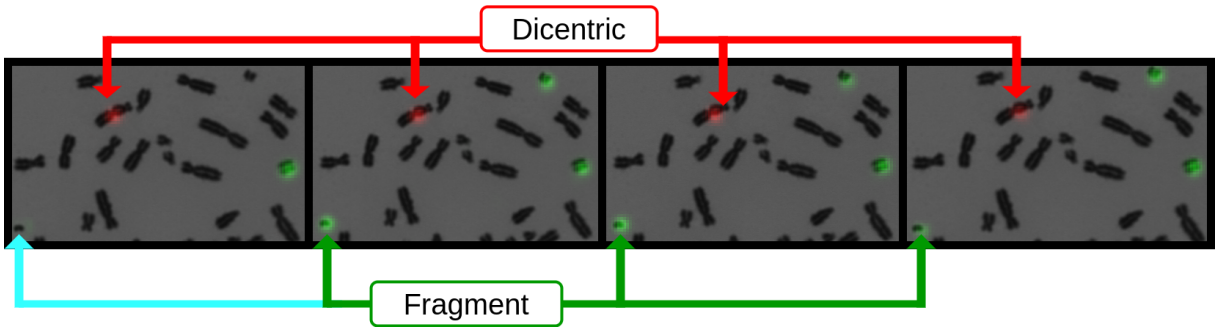


Figure 4.8 – Prediction diversity for a set of four different models. The first model does not predict the fragment in the bottom left corner (a very low confidence threshold would be required to consider it as a detection), but the three next model do. All four models predict the DC near the center of the image correctly.

In this section, we report the results obtained with the ensemble procedure. The parameter space of the model is explored through SGD. Although the selected checkpoints reach a similar validation performance, the predictions they yield vary, as shown by Figure 4.8. As mentioned earlier, this can be used to filter spurious predictions, as those are less likely to be present in all models of the ensemble.

Precision is a monotonously increasing function of voting and confidence thresholds, while Recall decreases with higher confidence and higher voting thresholds. End-users may fine-tune the balance between both of those metrics depending on their goal by choosing a specific (T_C, T_A) pair. To estimate the sensitivity of Precision and Recall to the sampling of the checkpoints, we evaluate those metrics for 100 samples ensembles and report the q_{05}, q_{95} interval for Precision and Recall in Figure 4.7).

To ensure that ensemble results are easily readable, we do not report surfaces for Precision and Recall. Instead, we select a single voting threshold and report the results over all agreement thresholds, like with the single model results. The ensemble provides a significant performance improvement over the single-model baseline; aggregation does help to filter out spurious detections and improves Precision and Recall, as reported in Tables 4.1 and 4.2. We chose the (T_C, T_A) parameters to keep Precision broadly similar across all models, for DCs and fragments. This makes Recall improvements more salient, but one could choose other values for confidence and vote thresholds. Overall, there is a wide set of threshold combinations that yield large performance improvements over the Metafer baseline. Furthermore, ensembling also reduces performance variation: the performance is closer between different ensembles than between single checkpoints. This suggests that our results are not dependent on a specific sampling or selection of the ensemble.

4.4.3 Robustness to non-chromosome objects in metaphase images

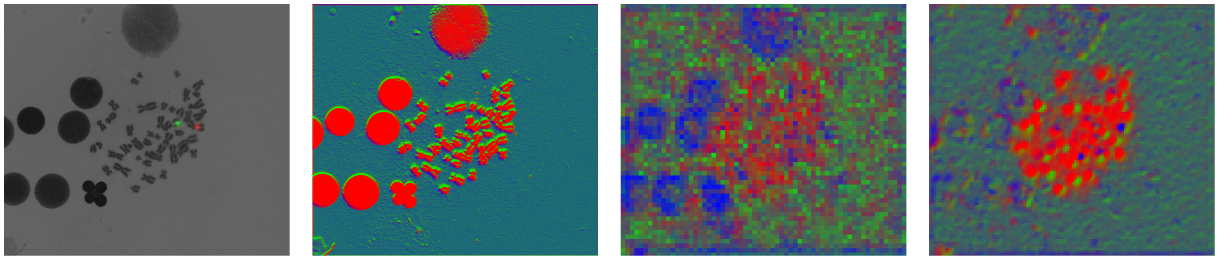


Figure 4.9 – Rejection of nuclei depending on model layer. First image from the left shows the input image and ground truth Gaussian heatmap. Second image shows PCA embedding of features at the output of the first encoder layer. Third image shows PCA embeddings of features at the last encoder layer. Rightmost image shows embeddings of features at the first decoder layer. The embedded feature maps are resized from H', W' to H, W so that every image has the same size.

An automated dosimetry system should always distinguish between chromosome and non-chromosome (nuclei, debris) objects. Current automated methods reject non-chromosome objects using explicit shape analysis. For example, a nuclei can be rejected using the fact that it is broadly circular and has a uniform texture. However, the shape of debris is usually more complex, which makes metaphase selection more difficult in most dosimetry

systems.

In our approach, we do not detect debris and nucleis explicitely, as they are not annotated in our dataset. Instead, Unet learns to reject debris from the training data, without the need for specific annotations or handcrafted object detection algorithms. In the rest of this section, we propose a simple visualization of this fact, using PCA.

For an image x_i and a set of parameters θ, the activation volume provided by the ℓ-th layer of the neural network is defined as:

$$f_i^\ell = \sigma(\theta_\ell f_i^{\ell-1} + b_\ell),\ f_i^\ell \in \mathbb{R}^{H' \times W' \times N_\ell}, \tag{4.14}$$

where N_ℓ the number of convolution filters in layer ℓ, i.e the number of channels of the output of this convolution layer. For a set of n images $\{x_1, \cdots, x_n\}$, the corresponding activation volume is of size $\mathbb{R}^{n \times H' \times W' \times N_\ell}$. This volume can be flattened in a matrix $E_\ell \in \mathbb{R}^{(n \times H' \times W') \times N_\ell}$, containing $n \times H' \times W'$ samples (rows) of a random vector of size N_ℓ. Once E_ℓ is centered and standardized, the eigenvectors of $E_\ell^T E_\ell$ form the usual PCA orthogonal basis $Q_\ell \in \mathbb{R}^{N_\ell \times N_\ell}$, so that $E_\ell Q_\ell^K$ (with $Q_\ell^K \in \mathbb{R}^{N_\ell \times K}$) projects each pixel of the activation volume (N_ℓ-sized vectors) onto the first K components (columns) of this basis.

By choosing $K = 3$ and normalizing the projected vectors to sum to one, we can plot a visualization of the activations in RGB space (Figure 4.9). Note that a different Q_ℓ is computed for each layer considered in Figure 4.9. Therefore, the colors do not have any specific meaning. The second image of Figure 4.9 from the left shows that the projection of feature vectors belonging to chromosomes and nuclei on the PC basis are highly similar, as the color of those regions is identical. The main reason is that the convolution filters in early layers have a small receptive field (see [120]) and mostly capture texture and edge information, which is close for subsets of chromosomes and nuclei. In the early stages of the encoder, non-chromosome objects are not differentiated from chromosomal objects.

The third and last images of Figure 4.9 from the left show that in the deepest layer of the encoder, the feature vectors belonging to nuclei and chromosomes are mapped to different principal components, corresponding respectively to blue and red pixels. By stacking convolutions the model aggregates information from a larger subset of the image (the receptive field increases), which is suitable to distinguish chromosomes from nuclei on the basis of their differences in shape. This is a first indication that our model learned to reject debris and nuclei without domain knowledge, which is further confirmed by our performance results.

4.4.4 Visualization of training trajectories

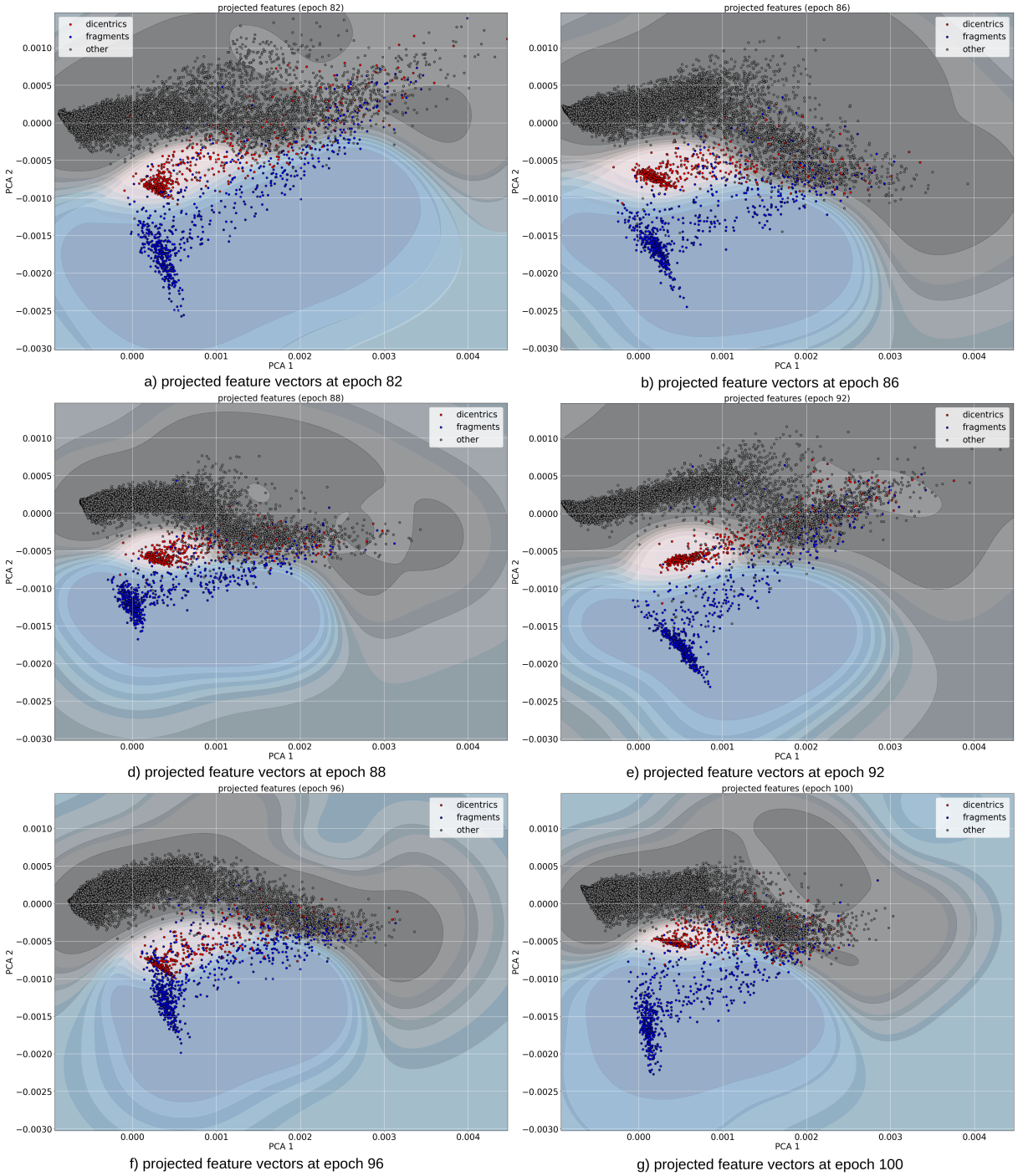
In this section, we discuss the visualizations produced by the method described in Section 4.2.4. In Figure 4.10, we see the latent space of Unet at 6 different epochs. Red points on the scatterplot represent locations containing DCs, blue points fragments and gray points are background locations. Because those snapshot are taken at regular intervals at the end of training, classes are well separated. We see that the decision boundary of the SVM classifier changes from timestep to timestep. Some locations remain well separated in latent space from others, but this is not true for all samples. Regions in the latent space where classes overlap correspond to areas of uncertainty.

While locations with high prediction uncertainty are hard to detect because neural networks tend to be overconfident [5], an ensemble of models can be used to retrieve this information. This fact is also visible in the latent space of Unet, as shown by Figure 4.11. In this Figure, we performed K-Means classification on our bag of features in the latent space for the last considered epoch in Figure 4.10. Once this is done, we plot the trajectories of the barycenters of those clusters over time. As we see, most cluster centers are well separated across epochs. This is reflected by the fact that if we consider the average of SVM classifiers across epochs, most feature vectors are in low-entropy regions. Feature vectors belonging to high-entropy regions correspond to uncertain detections can be filtered as described in Section 4.2.2.

4.4.5 Transfer between training and calibration curve datasets

As explained in Section , most images routinely analyzed by biologists do not contain any aberration, even for high doses. In the previous section, we saw that the model described in this Chapter performs very well in terms of object detection metrics. However, we still need to investigate wether a model trained on this dataset could accurate estimate aberration counts on a more realistic dataset.

Initial calibration curve estimates were unsatisfying: our ensemble would overestimate low doses, and underestimate high doses. Two additional details were needed to improve performance. First, we investigated the Cumulative Distribution Function (CDF) of the maximum probabilities predicted by the members of the ensemble for the DC and fragment class. Those CDFs were estimating using the 4 Gy subset of the calibration curve dataset, and are shown in Figure 4.12. Instead of setting a single threshold for all model, we picked a quantile and retrieved the corresponding CDF value for each model. Fur-



a) projected feature vectors at epoch 82

b) projected feature vectors at epoch 86

d) projected feature vectors at epoch 88

e) projected feature vectors at epoch 92

f) projected feature vectors at epoch 96

g) projected feature vectors at epoch 100

Figure 4.10 – Snapshot of the training trajectory in feature space. Each point of every scatterplot represents a fixed location in an image (DC, fragment or background). Because of the stochasticity of training, the corresponding feature vector moves in feature space. The contour map showcases the decision boundary of a kernel SVM classifier trained to predict the type of feature vector depending on its location in feature space.

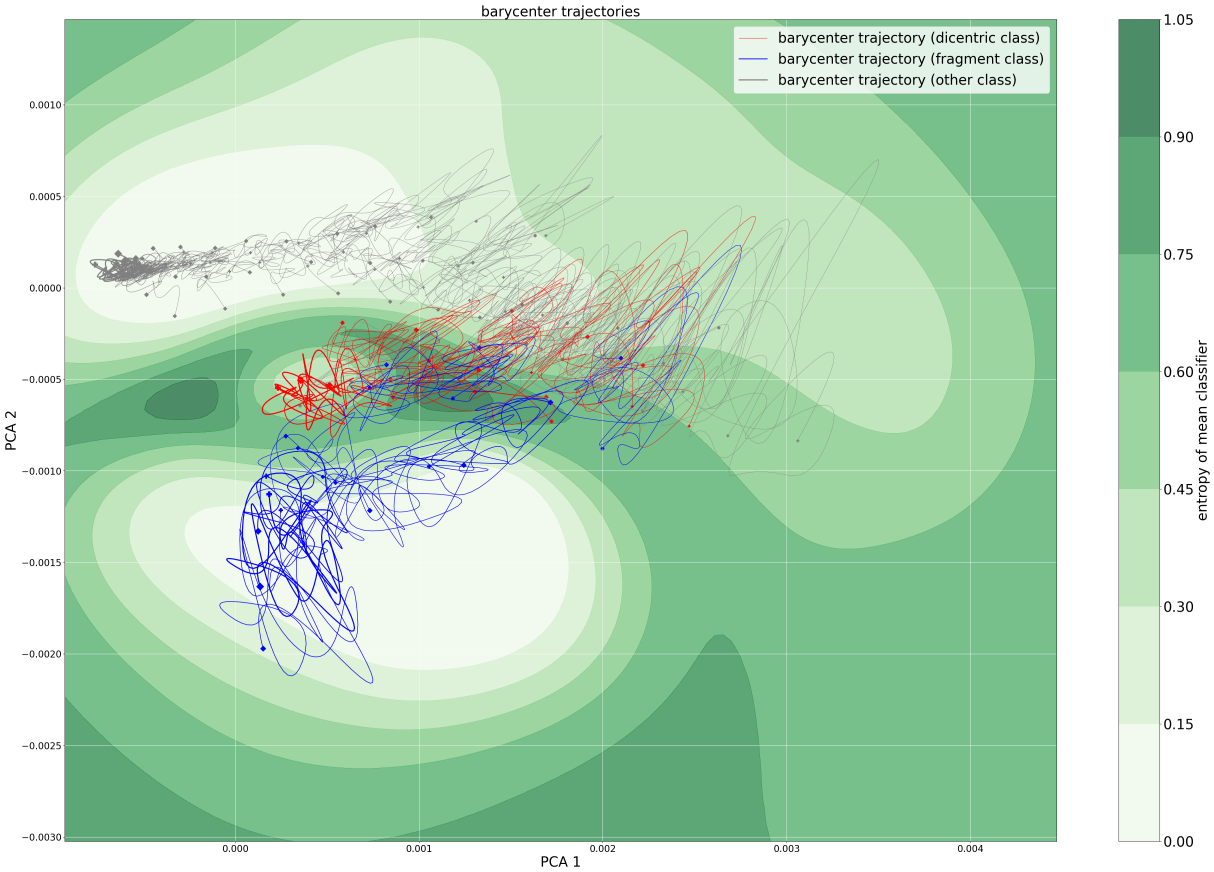


Figure 4.11 – Training trajectories of feature barycenters. The scatterplots displayed in Figure 4.10 are clustered with K-Means to simplify visualization. The trajectories of barycenters during training are displayed in this figure. The thickness of the trajectory shows the number of feature points in the barycenter.

thermore, we used domain knowledge to reject spurious DC detection. We considered DC detections if and only if at least one fragment was present in the same image. As usual in biological dosimetry, we fitted a linear-quadratic model to the point cloud of average DC count retrieved on the calibration curve dataset. This led to a large improvement in our calibration curve estimation, as shown in Figure 4.13. Furthermore, the Metafer calibration curve is semi-automated: DCs undergo a manual review, because of the very high FPR of the algorithm, as described in Section 4.4.1. On the contrary, our calibration curve is obtained in a fully automated setting.

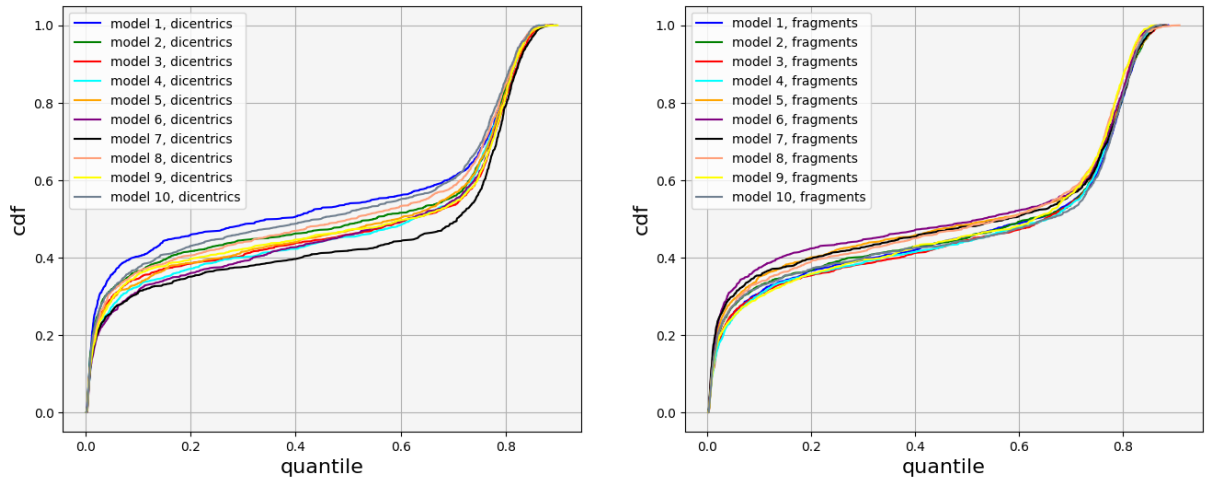


Figure 4.12 – Cumulative Distribution Functions (CDF) of the maximum probabilities predicted by every member of the ensemble for the DC (left) and fragment (right) class over all images corresponding to a 4 Gy dose in the calibration curve dataset.

4.5 Discussion and future works

In biological dosimetry, estimating the average number of chromosomal aberrations per peripheral blood lymphocyte metaphase is necessary to estimate an ionizing radiation dose. However, human expertise is required and is therefore a bottleneck to scale chromosome counting beyond a few hundred images per patient.

In this Chapter, we evaluated Unet as an aberration detection model for biological dosimetry. Unet outperformed Metafer (a current commercial solution) in terms of Precision and Recall by a wide margin. Our approach is learning-based and differs significantly from the current state of the art in terms of how much domain knowledge of chromosome morphology is incorporated. We demonstrated a high level of performance without the need for significant shape modelling. Feature visualization suggests that the model learns to reject debris and nuclei in an unsupervised manner, without the need for object-specific annotations for monocentric chromosomes or debris. Furthermore, a simple regularization term modelling the intrinsic heatmap sparsity helps performance.

We pushed this performance further by ensembling several checkpoints collected during training. We proposed a visualization of the latent features of Unet during training to explore the relationship between the dynamics of training and this performance improvement. Furthermore, we showed that the variation in performance between different (randomly sampled) ensembles is lower than between single checkpoints of a training run.

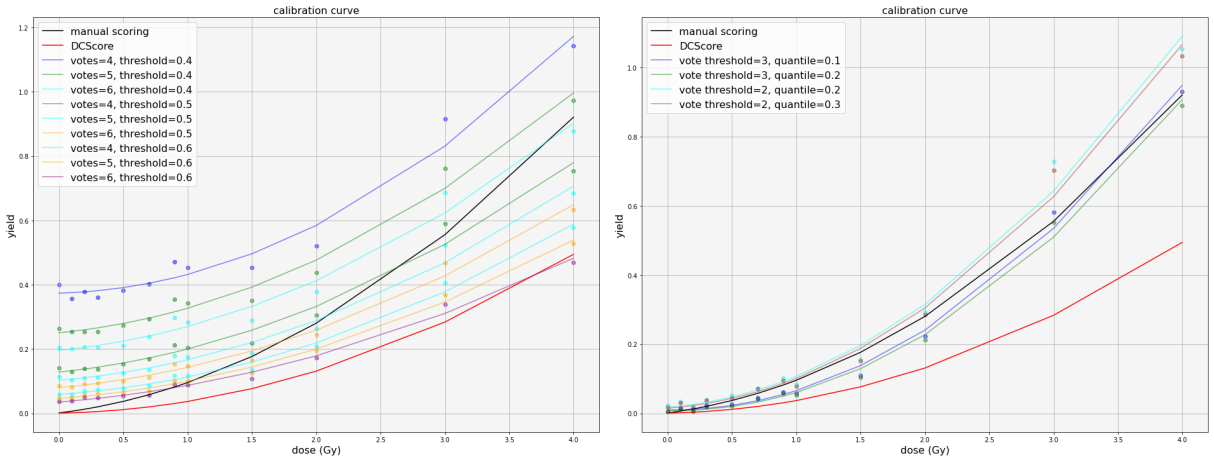


Figure 4.13 – Calibration curves estimated by the ensemble. Left: calibration curve before setting a threshold per model and using domain knowledge. Right: calibration curve after model-adaptive thresholding and using domain knowledge. To improve readability, we show the 4 curves closest to the manual calibration curve displayed in black. Metafer curve is displayed in red.

This is especially relevant in the context of the deployment of a deep learning model in an automated fashion in a medical setting. Those improvements can be achieved without the need for any architectural modifications or extensive hyperparameter calibration.

Our training dataset is not representative of real world data: the average number of aberration per cell is over 1, which corresponds to an extremely high dose of ionizing radiation. It is very likely that some of the spurious detections can be attributed to this training set imbalance. We evaluated our ensemble of Unet in a realistic setting, on a calibration curve dataset. Using model-adaptive thresholding and the domain knowledge of co-occurence of DCs and fragments, we reach a very competitive calibration curve, widely outperforming the Metafer baseline. Furthermore, our ensemble can be used in a fully automated fashion, while the Metafer solution requires manual review to reduce the number of false positives. It is therefore possible to build a competitive aberration detection system even with a large distribution gap between training and inference images.

Conclusion

Contributions

The main contribution of this thesis is a proof of concept based on deep learning for a chromosomal aberration detection model in cytogenetic biological dosimetry. This model relies on the Unet architecture. It is trained by minimizing a L_2 loss and a sparsity-promoting regularization term over a large, labelled dataset. We use the fact that gradient descent samples can be seen as approximate samples for the posterior distribution of the network to retrieve a diverse ensemble of models during training, and we build an interpretable agregation function for this ensemble. This function relies on two thresholds, a confidence thresholds and an agreement thresholds, which can be set by the end-user to balance between precision and recall.

In our labelled dataset, all images contain at least one dicentric chromosome, which means that aberrations are much more prevalent than in real life. This oversampling prevents the discovery of trivial models during training, where no aberration is every predicted. However, this also means that there is a distribution gap between a real blood sample and our training dataset. Using some domain knowledge, like the co-occurence of DCs and fragments, we reject a large number of spurious detections, and improve the false positive rate.

Using this domain knowledge, the counts provided at each dose match the manually estimated calibration curve very closely. Furthermore, unlike the previous semi-automated tool used at IRSN where semi-automatic, our solution achieves this false positive rate in a fully automated fashion.

Moreover, we provide a PCA-based visualization of the features of our model. This allows one to display the differences between the features corresponding to chromosomal objects and non-chromosomal objects. This rejection of non-chromosomal objects is achieved without the need for specific modelling or labelling. This contrasts with current commercial solutions that need to reject non-chromosomal objects based on their shape or texture during the chromosome detection step. The deep learning paradigm, where rich image features are learned on large datasets instead of modelled explicitely proved rele-

vant in this specific biomedical imaging application, in spite of the all the issues identified in Section 2.2.

Furthermore, we used PCA to provide a visualization of the training trajectories of Unet outputs. By plotting the trajectories of pixels corresponding to aberrations in feature space, we are able to evaluate how well our model separates aberrations from background during training. We trained several SVM classifiers, each corresponding to a different epoch, and therefore a different feature space. Visualizing the entropy of the distribution predicted by the average classifier allows one to justify our ensembling gains. Low entropy regions in feature space correspond to aberrations receiving a high number of votes, as most members of the ensemble agree on the object class. On the other hand, high entropy regions correspond to aberrations receiving a lower number of votes.

Perspectives

While our model reaches a high level of performance, both in terms of object detection metrics and calibration curve estimation, it may not be considered as a complete dose estimation pipeline. Because of variations in image quality some metaphases are completely unusable. Chromosomes may be underspread, or the image may be to blurry for accurate chromosome classification. Currently, both of our datasets (training and calibration curve estimation) are comprised of metaphases that are good enough for human scoring. A fully automated dose estimation pipeline needs a metaphase selection step. Metaphase selection with conventional computer vision techniques was explored in [89] to guarantee that every metaphase is complete, i.e., that it contains 23 pairs of chromosomes. More recently, a ConvNet was trained to perform metaphase selection, reaching over 99% accuracy [90]. However, the metaphase selection step does not segment chromosome explicitely and relies on a ResNet-based binary classifier instead.

Metaphase selection could be solved with several architectures mentioned in Chapter 1, like Faster R-CNN [9] or ResNet [8]. To ensure that metaphases are complete, Faster R-CNN is appropriate. However, this means that we need bounding box labels for every single chromosome which is time consuming. In [121], Pachitariu *et al.* proposed an active learning biomedical image segmentation software called Cellpose. The underlying segmentation model is trained on biomedical images, and predictions can be edited to re-train the model. This approach could be used to train a chromosome segmentation model without building a large labelled dataset.

Furthermore, while our training dataset is comprised of 5430 labelled images, this dataset is extracted from a larger database containing around 80 000 images. Our training subset is comprised of all images containing at least 1 aberration. Even if this larger archive does not contain any chromosomal aberration, it still brings a much larger repository of MC and cellular debris examples which would help MC-DC discrimination provided that we can design an effective pre-training task. It has been shown that because conventional similarity-based SSL techniques discard spatial information with global average pooling, they are less effective for pre-training dense models [122]. Therefore, designing pretext tasks for dense prediction models can be considered in future works. However, pretraining models on this large dataset would also be significantly more costly. The increased computation time suggests that the exploration of small architecture changes or new hyperparameter configurations is slowed down considerably, which makes designing such a pretext task more challenging.

Appendix A

Sparsity in optical flow for microfluidics imaging

From June to September of 2022, I stayed at the Engineering Physics department of McMaster University, in Canada. This research stay was supervised by Qiyin Fang and focused on assisting Tianqi Hong's PhD work by designing methods for optical flow estimation in microfluidics cell imaging. This optical flow estimation method uses the same sparse regularizer as the proof of concept presented in Chapter 4. This method is the subject of this Appendix.

Abstract

Building labelled optical flow datasets is very challenging, which makes the training and evaluation of those on unseen sequences difficult. To overcome this difficulty, optical flow models are often trained on benchmark datasets composed of synthetic image sequences. This means that during inference on unseen image sequences, there will be a distribution gap between the training and testing data. This distribution gap leads to errors if unseen data does not match the training distribution. This is often the case in biological, medical of satellite images, because those applications showcase unusual object sizes, viewpoints and levels of optical resolution or noise. In this context, a pre-trained, supervised model might fail to estimate a flow field where the flow discontinuities match the object boundaries for small objects. Furthermore, slow objects cannot be continuously detected by those pre-trained models, and they tend to be merged in the background.

The main context for this chapter is shadow imaging in microfluidics applications, where the issues mentioned above are especially relevant. Cells are very small with respect to the field of view, and showcase variations in speed. Because of this, the estimated flow discontinuities do not match cell boundaries, and slow cells are not detected. We evaluate boundary mismatch and speed dependence using the agreement between motion

field segmentation and cell masks as a proxy measurement of flow quality. Moreover, we propose a novel regularizer to adapt unsupervised optical flow models to microfluidics imaging.

A.1 Introduction

Lensless microscopy systems relying on cost-efficient Commercial-Off-The-Shelf (COTS) hardware are proposed as an alternative to flow cytometry methods [123]. The image sensor used for acquisition is mounted over the transparent microfluidics channel. The channel is backlit with incoherent lighting. Usually, the considered samples (red blood cells, parasites) have enough transparency for shadow imaging. The optical resolution of the device depends on the distance between the channel and the sensor, but cannot exceed twice the pixel size. Nevertheless, lensless devices have several advantages. They are cheaper than a flow cytometer, which is the current gold standard for fluid analysis in the medical field. They also offer a very wide field of view, which improves the *throughput* of the device. Figure A.1 provides a schematic of the device implemented in [123]. In return, sophisticated image processing is required to reproduce the functionality of flow cytometers, like cell identification or cell counting. Because cell rotation motion can be used to classify cells, estimating a motion field from pairs of images is a crucial step to build a COTS flow cytometer using microfluidics imaging.

Optical flow estimation was traditionally formulated as an optimization problem [124]–[127]. Variational methods generally penalized the norm of the gradient of the motion field to favor piecewise smoothness [124]. For several decades, variational methods were considered to be the most competitive methods for optical flow estimation [127]. However the errors generated by the pixel-wise data term are often compensated by an undesirable over-smoothing. An alternative approach consisted in fitting parametric motion models in the neighborhood of each pixel [128]. The motion field is estimated from a region-based data term that exploits the consistency of several pixels in the region. Despite their superior robustness to noise, these methods are usually unable to compete with global methods in terms of accuracy at low levels of noise [127], in various application fields (fluid mechanics [129], cell imaging and microscopy [130]–[133]). The reason is the difficulty to determine the boundaries of regions where the motion can be accurately approximated by a parametric motion model. However, it has been shown that when appropriate regions are chosen, local parametric models can yield good performance [134].

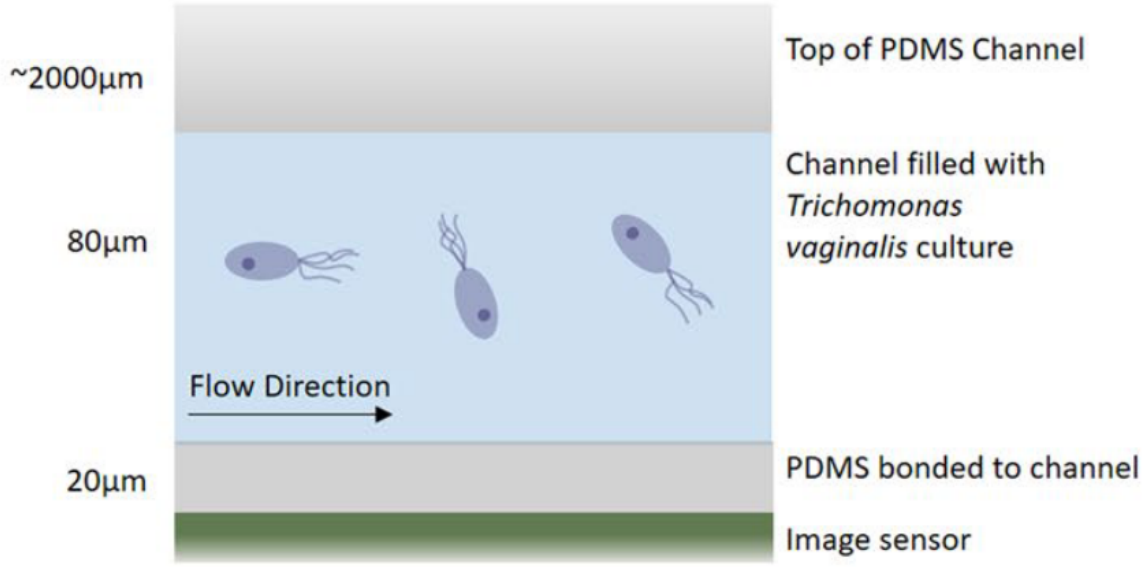


Figure A.1 – Schematic of the microfluidics device implemented in [123]. A $\sim$ 80 μm channel is machined in PDMS (gray). A fluid (light blue) carrying objects (cells, parasites) is pumped through the channel for analysis. An image sensor (in green) is bounded directly to the channel. The channel is lit from the opposite side to the image sensor.

More recently, convolutional neural networks have been shown to be very effective models in a wide variety of computer vision tasks, including optical flow estimation. The first example of supervised flow estimation using deep learning was FlowNet, which uses a Unet-like architecture [135]; it is trained using labelled datasets, usually built from synthetic video sequences; in this line of research, the current state-of-the-art method is Recurrent All-Pairs Field Transforms (RAFT) [136]. Several unsupervised methods were also published [137]–[139]. Those methods still use the basic components introduced by [128], but they use convolutional neural networks (or transformers) as non-linear feature learners. Furthermore, they consider specific training methods that leverage the advantages of deep architectures. Current deep-learning-based unsupervised optical flow estimation provides strong results [138], [140], even in cases where large motions or occlusions are present.

The remainder of this chapter is organized as follows. In Section A.2, we present the basics of unsupervised optical flow estimation with convolutional networks, with a special focus on regularized training losses. In Section A.3, we present the dataset, as

well as a detailed reasoning for our use of segmentation as a proxy measurement when ground truth optical flow is unavailable in the context of microfluidics imaging. Finally, we show qualitative and quantitative results for the competing models also introduced in this section.

A.2 Unsupervised CNN-based method for sparse optical flow estimation

A.2.1 Definition of data term and loss function

For a pair of consecutive grayscale images $I_1, I_2 \in [0,1]^{H\times W}$, let us consider a vector field f_{12}, where at each location $(u,v) \in \Omega = \{0,\cdots,H\} \times \{0,\cdots W\}$ we have $f_{12}(u,v) = (\Delta_u, \Delta_v)$ so that $I_2(u+\Delta_u, v+\Delta_v) = I_1(u,v)$. This vector field quantifies the displacement of pixels. Note that the displacement is assumed to be small to derive linear equations with respect to image displacement. This is usually correct if the timesteps are small enough, *i.e* the framerate is high enough.

If the flow f_{12} was perfectly estimated, we would be able to reconstruct I_2 from I_1 by resampling I_1 at the new locations given by the flow field. This resampling step is defined as a warping function:

$$w(I_1, f_{12}) = I_2. \tag{A.1}$$

The reconstruction of I_2 from I_1 and the estimated flow field f_{12} provides a natural criterion to evaluate the quality of the estimated flow field, usually called the photometric consistency criterion:

$$\mathcal{D}(I_1, I_2, f_{12}) = \sum_{u,v\in\Omega} \rho\left(I_2(u,v), w(I_1, f_{12})\right), \tag{A.2}$$

where ρ is a robust loss criterion (*e.g.* $\rho(x) = \sqrt{x^2+\varepsilon^2}$). In general, variational methods minimize a cost function directly with respect to f_{12} [127]. In deep-learning-based approaches, the major difference is that the training loss is minimized with respect to the estimator parameters. In our case, $f_{12} = g_\theta(I_1, I_2)$ where g_θ, which is a deep convolutional neural network parameterized by a set of weights θ. Given a dataset of image pairs $\{(I_1,I_2)_1, \cdots, (I_1,I_2)_n\}$, the model parameters are obtained by minimizing the following

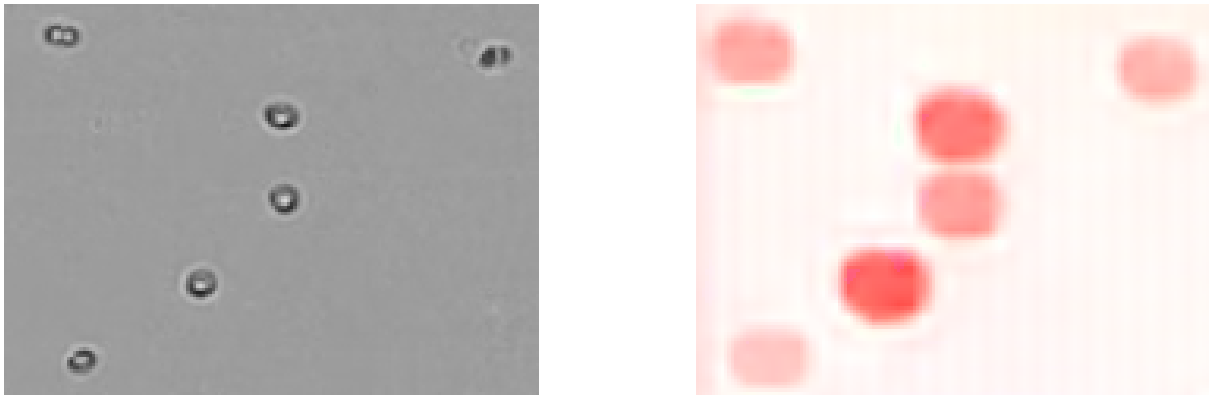


Figure A.2 – Example of smearing in unsupervised optical flow estimation. Left: input grayscale image, Right: flow field estimated by UFlow without any regularization. Bottom left: HSV wheel corner indicating how the flow map (bottom) should be interpreted. Color indicates flow orientation, while saturation indicates flow norm.

criterion as follows:

$$\theta^\star = \arg\min_\theta \sum_{i=1}^{n} \sum_{u,v} \rho\left(I_2(u,v), w(I_1, g_\theta(I_1, I_2)(u,v))\right). \tag{A.3}$$

Because of the non-linearities of g_θ, this training loss is not convex, and finding a global minimum is not feasible. Accordingly, $\theta^\star$ usually corresponds to a local minima of the loss function.

A.2.2 Definition of spatial regularizers

Because of occlusions near object boundaries, matching a pixel between I_1 and I_2 may be very difficult. A single pixel in I_1 might have several low-error (in terms of intensity) matches in I_2. Because of this, unsupervised flow estimates often exhibit "smear", that is the flow boundaries do not match object boundaries, as shown in Figure A.2. As the data fitting may not be informative, regularization is used to impose some prior. The following *1^{st}-order smoothness* is commonly used in optical flow to adapt to object boundaries:

$$\mathcal{R}(f_{12}, I_1) = \beta \left[\sum_{u,v} \|\nabla_{u,v} f_{12}\|_1 e^{-\alpha|\nabla_{u,v} I_1(u,v)|}\right], \tag{A.4}$$

where α modulates edge weighting, and

$$\nabla_{u,v} f_{12} = \begin{bmatrix} f_{12}(u,v) - f_{12}(u+1,v) \\ f_{12}(u,v) - f_{12}(u,v+1) \end{bmatrix}. \tag{A.5}$$

Note that this is a simple linear transformation of the image so that the computational overhead is minimal. While this criterion forces the alignment of image and flow boundaries, it is unsatisfying because while object edges should also be flow boundaries, not all edges are object edges. Objects with internal edges can exhibit over-regularized flow with this regularizer.

In the context of microfluidics imaging, we know *a priori* that objects are small and exhibit relatively small motion. The background, which is most of the surface of the image, is expected to exhibit zero flow. Therefore, the flow field is expected to be sparse, that is the number of locations where the flow is non-zero is very low. Therefore, we can consider the following sparse variation regularizer, introduced in [11]:

$$\mathcal{R}_{SV}(f_{12}) = \lambda \left[\sum_{(u,v)} \rho^2 \|\nabla_{u,v} f_{12}\|_2^2 + (1-\rho)^2 f_{12}^2(u,v) \right]. \tag{A.6}$$

The first term encourages a smooth flow field, while the f_{12}^2 term controls the sparsity of the flow. The two terms are balanced with the hyperparameter ρ, usually $\rho \simeq 0.1$. If $\rho = 1$, (A.6) is nothing else than the popular Tikhonov regularizer. In [11], the square root of this regularization term was taken to ensure convexity, which is not relevant in our training framework, as the non-linearities of g_θ makes the training objective non-convex.

In our implementation of Sparse Variation, we found that it is necessary to regularize the flow estimate at all layers of the feature pyramid of UFlow (see Section A.3.3 for details of UFlow). However, as the criterion in (A.3) is a sum over all pixels, the hyperparameter λ in (4.3) needs to be adjusted at each scale to ensure it scales with the magnitude of the loss. As the height and width are divided by a factor 2 at each layer, we multiply λ by a factor 4, and we optimize the average regularization over all layers:

$$\mathcal{R}_{SVA}(f_{12}) = \frac{1}{L} \sum_{\ell=1}^{L} 4^\ell \lambda \mathcal{R}_{SV}(f_{12}^\ell), \tag{A.7}$$

where f_{12}^ℓ the estimated flow at layer ℓ.

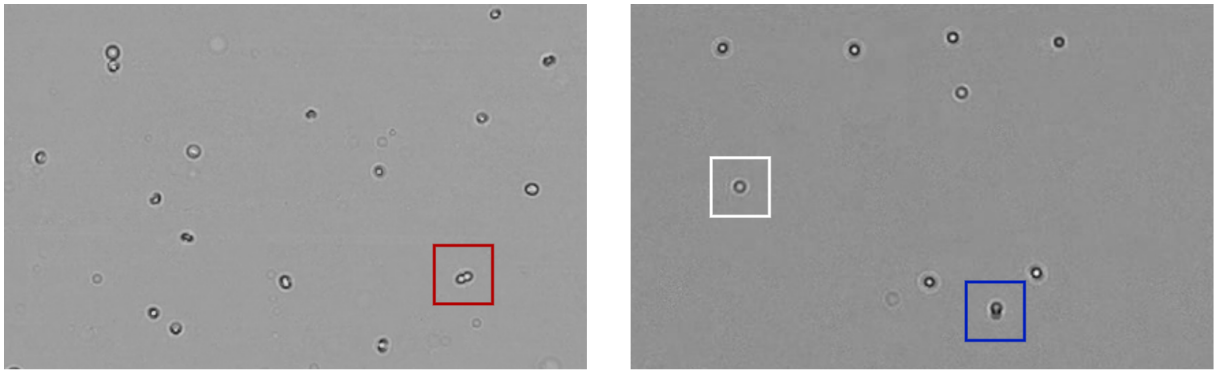


Figure A.3 – Single frame of the two sequence dataset used in this chapter. Left: crop of a frame from the red blood cell (RBC) sequence. A red blood cell is contained in the red square. Right: crop of a frame from the polystyrene and *yeast* sequence. A polystyrene bead is localized in the white square and a yeast in the blue square.

A.3 Experimental results and comparison of supervised and unsupervised methods

A.3.1 Description of datasets

Our experiments were performed on several sequences acquired through shadow imaging in a microfluidics channel. In the two sequences used for illustration in this Section, each frame is of size $W = 1920, H = 1080$. In the first image sequence (*yeast* sequence) composed of 586 frames, the fluid carries polystyrene beads and yeast through the microfluidics channel. In the second image sequence (RBC sequence) comprised of 400 images, a blood sample is imaged, also with shadow imaging. The density of cells is much higher in the RBC sequence, which makes estimating flow boundaries accurately challenging. In Figure A.3, crops from each sequence are shown. As the camera is not moving and the objects are relatively small (around 10 by 10 pixels in a FullHD image), we estimated the background (for background subtraction) using a simple average of the sequence. In what follows, we only consider the background subtracted sequences.

A.3.2 Evaluation metrics for optical flow estimation

In optical flow benchmark datasets, the ground truths are generally available [127]. If this is not the case, flow fields can be evaluated qualitatively. However, this might not be an objective way to compare several optical flow estimation methods. To provide

a fair comparison, we built segmentation annotation for the *yeast* and RBC sequences with Ilastik interactive segmentation tool [141]. Those annotations are used to build a proxy quality measurement for optical flow, focusing on the difference between flow and object boundaries. This is achieved by building a segmentation map of our optical flow estimation, and using conventional image segmentation metrics like Intersection Over Union (IoU).

To separate cells from the background using an optical flow estimate (*i.e.*, to build a segmentation map), we chose to threshold the L_2 norm of the flow field. We translated the histogram of the norm to the $[0, 1]$ interval to ensure that thresholds are comparable from frame to frame. Given that the speed of the cells varies during the sequence (especially for the RBC sequence), we estimated a new threshold for each frame using the Otsu method [80].

Moreover, the speed variation of cells inside a microfluidics channel may complicate the evaluation procedure. While background subtraction ensures that there is no truly stationary cell in the dataset, very slow cells might be estimated as background. Therefore, some segmentation errors are due to slow cells which are missing completely (*i.e.*, every pixel of the cell is predicted to be a static, background pixel), and others are caused by imperfect flow discontinuities. Because of this, we split the segmentation error into two parts and compute two IoUs. For each cell in the true segmentation map, we check if there is a non-zero intersection with the segmented flow map. If this is not the case, the pixels of this cell are labelled as background. Here, $IoU_{boundary}$ is computed between this new true segmentation map and the flow segmentation map. $IoU_{boundary}$ corresponds exclusively to cell boundary estimation error. IoU_{all} is the IoU between the original segmentation map and the segmented flow map. Therefore, $IoU_{boundary} > IoU_{all}$. This decomposition allows one to verify if a model reaches a better IoU because it detects slow cells better, or because it provide a better estimation of cell boundaries.

Finally, we also evaluate the model by comparing cell counts, which gives an indication of how well the model is able to separate close cells. If adjacent cells are insufficiently separated, the global cell count is underestimated. On the other hand, small segmentation artifacts leads to an overestimated cell count.

A.3.3 Description of competing methods

In our experiments, We compared two different architectures. First, we choose RAFT [136] as the baseline method for supervised optical flow estimation, because of its high

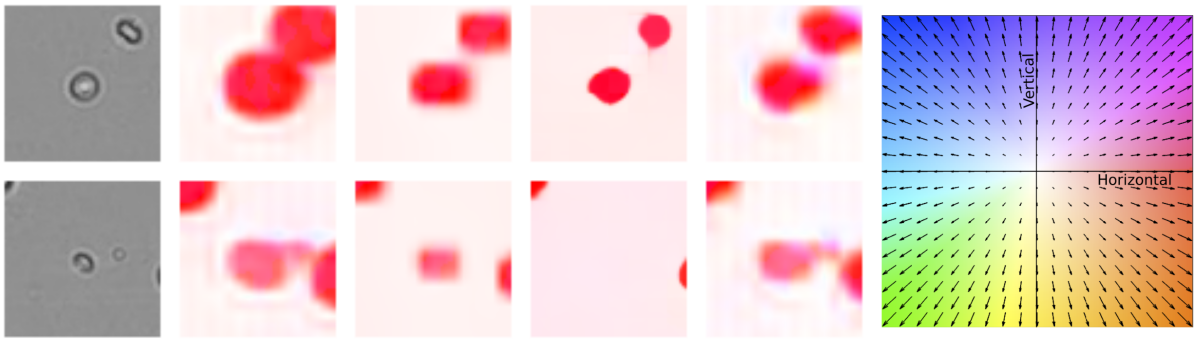


Figure A.4 – Comparison of all flow models evaluated in the chapter. First column is a random sample of cells in the sequence, retrieved at the same timestep (first image of the RBC sequence). Second column is the unregularized UFlow model. Third column is UFlow regularized with first-order smoothness. Fourth column is RAFT, and last column is UFlow regularized with Sparse Variation.

performance and availability in the `torchvision` package. We did not modify RAFT, and used the pre-trained weights available in `torchvision`. Second, we considered UFlow [137] as it is a performant and flexible unsupervised optical flow estimation architecture. UFlow [137] consists in a feature pyramid, where both images are repeatedly downsampled, and progressively deeper features are built using convolutional layers. At each layer, a cost volume is computed [135], and an optical flow field is estimated. In [137], the final flow field is estimated at 1/4 of the input resolution and then upsampled to compute the photometric loss. Nevertheless, we made some modifications compared to the original UFlow implementation to better fit our case study. Because dense mention is crucial to accurately estimate cell rotation, and therefore to identify cells, optical flow is estimated at the highest possible resolution, instead of 1/4 of the input resolution. We did not use bidirectional losses, occlusion estimation or self-supervision to keep training as simple as possible. In the end, three models are trained: UFlow without any regularization, UFlow regularized with a Sparse Variation criterion ($\rho = 0.1$ and $\lambda = 2 \times 10^{-10}$) and UFlow regularized with an edge-alignment criterion ($\beta = 0.1$ and $\alpha = 100$). We ran training for 25 epochs on the RBC sequence, using a batch size of 2, and a learning rate of 0.0001.

A.3.4 Experimental results

Qualitative evaluation Figure A.4 shows a qualitative comparison of the different flow estimation methods. In the flow field estimated by RAFT, some cells that exhibit positive

Figure A.5 – Cell merging and fading issues in the flow field estimated by RAFT on the RBC sequence. Four consecutive crops (frames 387, 390, 393 and 396) are displayed. Although two cells are present, the boundaries estimation are only correct if the the two cells are far enough.

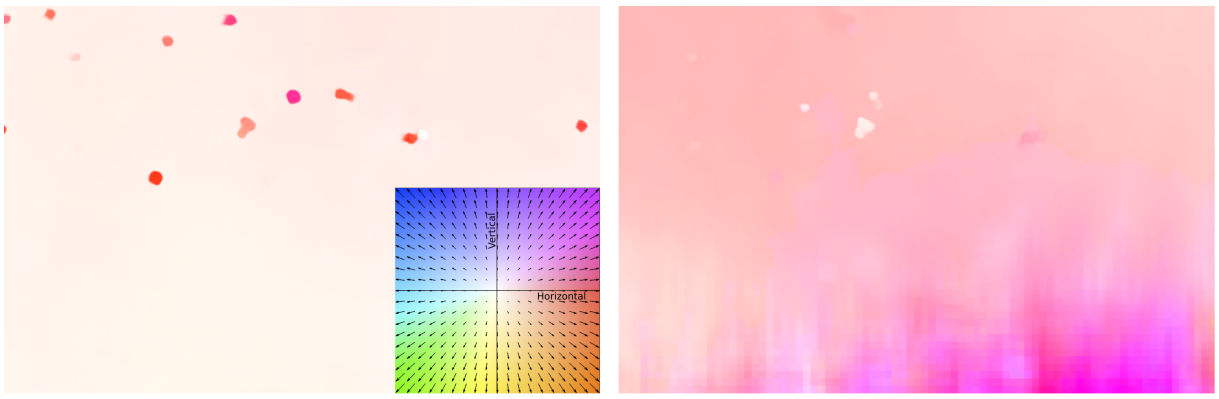


Figure A.6 – Crops of frames 278 (top) and 279 (bottom) of the flow field estimated by RAFT on the RBC sequence.

flow in frame t will exhibit zero flow in frame $t+1$. The cell is effectively "switched off". While a stationary cell should exhibit 0 flow even though the pixel intensities are non-zero, the instantaneous switch background and foreground for specific cells is not realistic. Slowing cells should exhibit a progressively decreasing average flow norm. This also prevents long-term tracking of cells, which is especially problematic because long-term rotation patterns are a crucial feature for cell classification, as shown in [123]. Furthermore, Figure A.5 shows that RAFT sometimes merges cells that are close to each other. This prevents accurate cell counting, and makes cell identification harder. In Figure A.4, we see that 1^{st}-order smoothness tends to produce flow fields with well separated cells. Nevertheless, its worth noting that this regularizer generates unrealistic square cell borders.

Unsupervised models trained directly on the RBC sequence mitigate the issue of disappearing cells. However, the flow and cell boundary do not match as accurately as with RAFT. Regularization seems to improves this shortcoming.

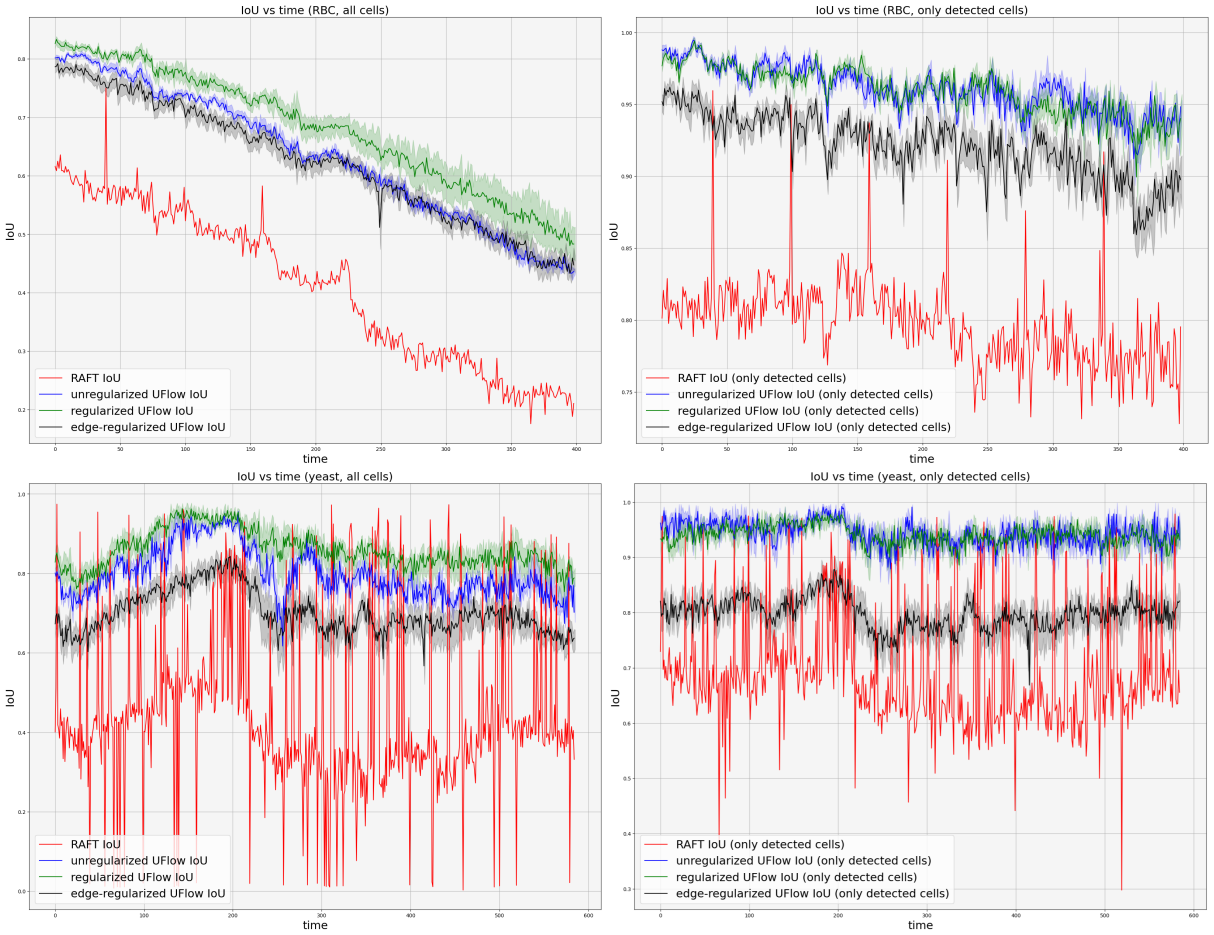


Figure A.7 – IoU as a function of time. Top row is IoUs over time for RBC sequence. Bottom row is IoUs over time for yeast sequence. Left column is IoU for all cells, right column is IoU for detected cells only, as described in Section A.3.2. Four models are compared: RAFT (red), UFlow without any regularization (blue), UFlow with Sparse Variation regularization (green) and UFlow with edge-alignment regularization (black). Shaded areas indicates the (q_5, q_{95}) interval of counting performance over the last 5 epochs of training, for all models except RAFT, where only the last epoch weights were available in `torchvision`.

Quantitative evaluation of boundary estimation In Figure A.7, we see the results for the *yeast* sequence on the bottom row. The Sparse Variation model achieves the highest IoU for all cells. Interestingly, if we only consider *detected* cells, the performance gap with the unregularized model is very low which is also true for the RBC sequence, as shown in the top row of Figure A.7. This suggests most of the variation in terms of IoU across time is due to disappearing cells.

The variation of IoU across time is much higher for the RBC sequence than for the

yeast sequence, like for counting performance in Figure A.8. As explained before, this performance can be decomposed into an IoU for detected cells, and an IoU for all cells. Thanks to this decomposition, we can confirm that disappearing cells are the largest reason for this drop in IoU across time, as the IoU for detected cells is consistently much higher across time.

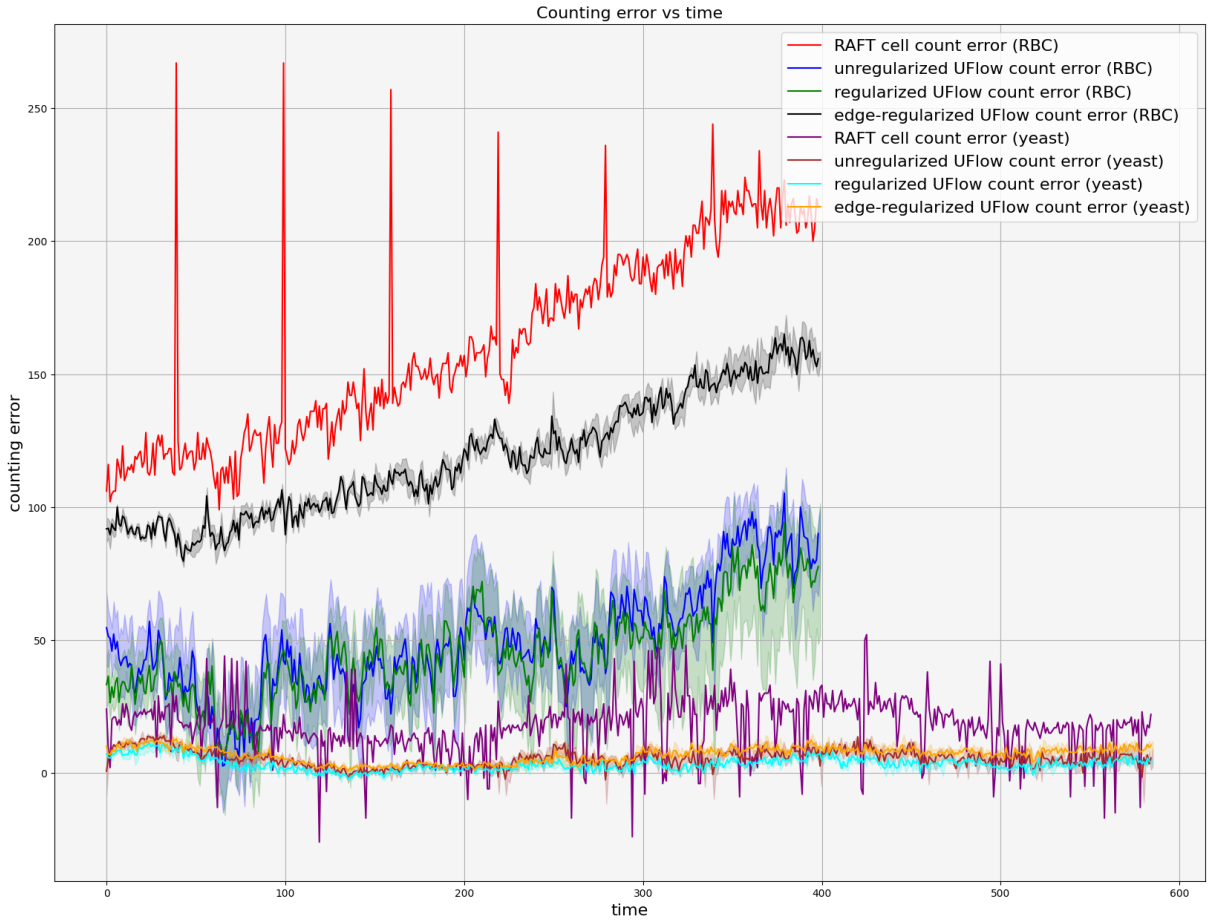


Figure A.8 – Counting error as a function of time. The error for both sequences is represented on the same figure, but both sequences are not the same length. The RBC sequence (red, black, blue and green lines) is shorter than the *yeast* sequence. Four models are compared: RAFT (red), UFlow without any regularization (blue), UFlow with Sparse Variation regularization (green) and UFlow with edge-alignment regularization (black). Shaded areas indicates the (q_5, q_{95}) interval of IoU over the last 5 epochs of training, for all models except RAFT, where only the last epoch weights were available in `torchvision`.

Quantitative evaluation of cell counting As we see in Figure A.8, cell counting results are heavily speed-dependent, especially in the RBC sequence, although some architectures fare better than others. RAFT produces the highest overestimation of the number of cells. Worse, this overestimation is inconsistent, with very high peaks. Those peaks correspond to frames where RAFT produces high-error outputs, as displayed in Figure A.6. In those frames, the optical flow estimation is over-smoothed, with a very strong background motion. Furthermore, the cells are not visible anymore.

UFlow produces a much more accurate cell count, especially for two models: the unregularized model (in blue in Figure A.7), and the model regularized with Sparse Variation (in green). While the unregularized model produces slightly better results for the RBC sequence, the variation in IoU across training epochs (materialized by the shaded confidence intervals) is much higher.

In the *yeast* sequence, we see that the counting error shows less variation across time, because of the constant speed of the cells in the channel. The Sparse Variation model achieves the lowest counting error, and the lowest variability across training epochs.

A.4 Conclusion

This chapter shows the limitations of pre-trained, supervised optical flow models for analyzing the modetion of small objects in a microfluidics setting. Because of the distribution gap between the training and testing dataset, the estimation of an accurate flow field for slow cells is unsatisfying. Cells tend to disappear unexpectedly, even though cells are not stationnary.

Unsupervised optical flow estimation adresses this issue, but the discontinuities of the flow do not match the object boundaries. While edge-alignment regularization is used very often in optical flow to adress this issue, it performs worse that our novel Sparse Variation regularization. The model trained with Sparse Variation regularization achieves the best performance (sometimes tied with the unregularized model), and the lowest variation in terms of IoU and cell counting performance over training epochs.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, « ImageNet Classification with Deep Convolutional Neural Networks », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012.

[2] O. Russakovsky, J. Deng, H. Su, *et al.*, « ImageNet Large Scale Visual Recognition Challenge », *International Journal of Computer Vision*, vol. 115, pp. 211–252, 2015.

[3] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, « Microsoft COCO: Common Objects in Context », *in European Conference on Computer Vision (ECCV)*, 2014, pp. 740–755.

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, « The Pascal Visual Object Classes (VOC) Challenge », *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2010.

[5] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, « On Calibration of Modern Neural Networks », *in International Conference on Machine Learning (ICML)*, 2017, pp. 1321–1330.

[6] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, « A simple baseline for bayesian uncertainty in deep learning », *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[7] S. Mandt, M. Hoffman, and D. Blei, « Stochastic Gradient Descent as Approximate Bayesian Inference », *Journal of Machine Learning Research*, vol. 18, 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun, « Deep Residual Learning for Image Recognition », *in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[9] S. Ren, K. He, R. Girshick, and J. Sun, « Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.

[10] O. Ronneberger, P. Fischer, and T. Brox, « U-Net: Convolutional Networks for Biomedical Image Segmentation », en, *in Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015, pp. 234–241.

[11] S. Prigent, H.-N. Nguyen, L. Leconte, *et al.*, « SPITFIR(e): a supermaneuverable algorithm for fast denoising and deconvolution of 3D fluorescence microscopy images and videos », *Scientific Reports*, vol. 13, p. 1489, 2023.

[12] J. B. Diederik Kingma, « Adam: A Method for Stochastic Optimization », *in International Conference on Learning Representations (ICLR)*, 2015.

[13] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. Wilson, « Averaging Weights Leads to Wider Optima and Better Generalization », 2018.

[14] I. A. E. Agency, « Cytogenetic Dosimetry: Applications in Preparedness for and Response to Radiation Emergencies », International Atomic Energy Agency, Text, 2011, p. 1.

[15] R. Balestriero, M. Ibrahim, V. Sobal, *et al.*, *A Cookbook of Self-Supervised Learning*, arXiv:2304.12210 [cs], 2023.

[16] C. Cortes and V. Vapnik, « Support-vector networks », *Machine Learning*, vol. 20, pp. 273–297, 1995.

[17] N. Dalal and B. Triggs, « Histograms of Oriented Gradients for Human Detection », *in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 886–893.

[18] D. G. Lowe, « Distinctive Image Features from Scale-Invariant Keypoints », *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[19] M. Abadi, P. Barham, J. Chen, *et al.*, « TensorFlow: a system for large-scale machine learning », *in USENIX conference on Operating Systems Design and Implementation*, 2016, pp. 265–283.

[20] A. Paszke, S. Gross, F. Massa, *et al.*, « PyTorch: An Imperative Style, High-Performance Deep Learning Library », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[21] G. Cybenko, « Approximation by superpositions of a sigmoidal function », *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989.

[22] A. Botev, H. Ritter, and D. Barber, « Practical Gauss-Newton Optimisation for Deep Learning », *in International Conference on Machine Learning (ICML)*, 2017, pp. 557–565.

[23] H. Ritter, A. Botev, and D. Barber, « A Scalable Laplace Approximation for Neural Networks », *in International Conference on Learning Representations (ICLR)*, 2022.

[24] J. Martens and R. Grosse, « Optimizing neural networks with Kronecker-factored approximate curvature », *in International Conference on Machine Learning (ICML)*, 2015, pp. 2408–2417.

[25] H. Robbins and S. Monro, « A Stochastic Approximation Method », *The Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.

[26] S. L. Smith, B. Dherin, D. Barrett, and S. De, « On the Origin of Implicit Regularization in Stochastic Gradient Descent », *in International Conference on Learning Representations (ICLR)*, 2020.

[27] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, « Understanding deep learning requires rethinking generalization », *in International Conference on Learning Representations (ICLR)*, 2016.

[28] K. Simonyan, A. Vedaldi, and A. Zisserman, « Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps », *in International Conference on Learning Representations (ICLR)*, Y. Bengio and Y. LeCun, Eds., 2014.

[29] L. A. Gatys, A. S. Ecker, and M. Bethge, « Image Style Transfer Using Convolutional Neural Networks », *in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2414–2423.

[30] D. Ulyanov, A. Vedaldi, and V. Lempitsky, « Deep Image Prior », *International Journal of Computer Vision*, vol. 128, pp. 1867–1888, 2020.

[31] A. Rosenfeld and J. K. Tsotsos, « Intriguing Properties of Randomly Weighted Networks: Generalizing While Learning Next to Nothing », *in Conference on Computer and Robot Vision (CRV)*, 2019, pp. 9–16.

[32] J. Frankle, D. J. Schwab, and A. S. Morcos, « Training BatchNorm and Only BatchNorm: On the Expressive Power of Random Features in CNNs », *in International Conference on Learning Representations (ICLR)*, 2020.

[33] S. d'Ascoli, L. Sagun, G. Biroli, and J. Bruna, « Finding the Needle in the Haystack with Convolutions: on the benefits of architectural bias », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[34] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction* (Springer series in statistics), 2nd ed. New York, 2009.

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, « Generative Adversarial Nets », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, 2014.

[36] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, arXiv:1312.6114 [stat.ML], 2022.

[37] T. Karras, M. Aittala, S. Laine, *et al.*, « Alias-Free Generative Adversarial Networks », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021, pp. 852–863.

[38] M. Arjovsky and L. Bottou, « Towards Principled Methods for Training Generative Adversarial Networks », *in International Conference on Learning Representations (ICLR)*, 2016.

[39] A. Radford, L. Metz, and S. Chintala, « Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks », *in International Conference on Learning Representations (ICLR)*, 2016.

[40] I. J. Goodfellow, J. Shlens, and C. Szegedy, « Explaining and Harnessing Adversarial Examples », *in International Conference on Learning Representations (ICLR)*, Y. Bengio and Y. LeCun, Eds., 2015.

[41] M. Mirza and S. Osindero, *Conditional Generative Adversarial Nets*, arXiv:1411.1784 [cs.LG], 2014.

[42] L. Mescheder, A. Geiger, and S. Nowozin, « Which Training Methods for GANs do actually Converge? », arXiv:1801.04406 [cs.LG], 2018.

[43] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, « Image-to-Image Translation with Conditional Adversarial Networks », *in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5967–5976.

[44] P. Ghosh, M. S. M. Sajjadi, A. Vergari, M. Black, and B. Scholkopf, « From Variational to Deterministic Autoencoders », arXiv:1903.12436 [cs.LG], 2019.

[45] G. E. Hinton and R. Zemel, « Autoencoders, Minimum Description Length and Helmholtz Free Energy », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 6, 1993.

[46] I. Higgins, L. Matthey, A. Pal, *et al.*, « Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework », *in International Conference on Learning Representations (ICLR)*, 2016.

[47] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, « Spectral Normalization for Generative Adversarial Networks », *in International Conference on Learning Representations (ICLR)*, 2018.

[48] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, « Momentum Contrast for Unsupervised Visual Representation Learning », *in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9726–9735.

[49] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, « A Simple Framework for Contrastive Learning of Visual Representations », *in International Conference on Machine Learning (ICML)*, 2020, pp. 1597–1607.

[50] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, « Unsupervised Feature Learning via Non-parametric Instance Discrimination », *in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, 2018, pp. 3733–3742.

[51] Y. LeCun, C. Cortes, and C. Burges, *MNIST handwritten digit database*. [Online]. Available: `http://yann.lecun.com/exdb/mnist/`.

[52] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, « Deep Clustering for Unsupervised Learning of Visual Features », *in European Conference on Computer Vision (ECCV)*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., 2018, pp. 139–156.

[53] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, « Unsupervised Learning of Visual Features by Contrasting Cluster Assignments », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 9912–9924.

[54] P. A. Knight, « The Sinkhorn–Knopp Algorithm: Convergence and Applications », *SIAM Journal on Matrix Analysis and Applications*, vol. 30, pp. 261–275, 2008.

[55] X. Chen and K. He, « Exploring Simple Siamese Representation Learning », *in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 15 750–15 758.

[56] E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig, « Laplace Redux - Effortless Bayesian Deep Learning », *in Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 20 089–20 103.

[57] B. Lakshminarayanan, A. Pritzel, and C. Blundell, « Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[58] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, « Snapshot Ensembles: Train 1, Get M for Free », *in International Conference on Learning Representations (ICLR)*, 2016.

[59] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson, « Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[60] D. Ruppert, « Efficient Estimations from a Slowly Convergent Robbins-Monro Process », Tech. Rep., 1988.

[61] B. Polyak, « New stochastic approximation type procedures », *Avtomatica i Telemekhanika*, vol. 7, pp. 98–107, 1990.

[62] S. Hochreiter and J. Schmidhuber, « Flat Minima », *Neural computation*, vol. 9, pp. 1–42, 1997.

[63] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, « On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima », *in International Conference on Learning Representations (ICLR)*, 2016.

[64] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, « Bayesian model averaging: a tutorial », *Statistical Science*, vol. 14, *4*, pp. 382–401, 1999.

[65] R. Girshick, « Fast R-CNN », *in IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[66] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, « Selective Search for Object Recognition », *International Journal of Computer Vision*, vol. 104, pp. 154–171, 2013.

[67] X. Zhou, D. Wang, and P. Krähenbühl, *Objects as Points*, arXiv:1904.07850 [cs.CV], 2019.

[68] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, « Focal Loss for Dense Object Detection », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 318–327, 2020.

[69] H. Law and J. Deng, « CornerNet: Detecting Objects as Paired Keypoints », *International Journal of Computer Vision*, vol. 128, pp. 642–656, 2020.

[70] R. Bayley, A. Carothers, X. Chen, *et al.*, « Radiation dosimetry by automatic image analysis of dicentric chromosomes », *Mutation Research/Environmental Mutagenesis and Related Subjects*, vol. 253, pp. 223–235, 1991.

[71] L. Roy, M. Delbos, N. Paillole, V. Durand, and P. Voisin, « Comparaison de systèmes d'analyse d'images cytologiques en dosimétrie biologique », *Radioprotection*, vol. 38, pp. 323–340, 2003.

[72] L. Roy, I. Sorokine-Durm, and P. Voisin, « Comparison between fluorescence in situ hybridization and conventional cytogenetics for dicentric scoring: a first-step validation for the use of FISH in biological dosimetry », *International Journal of Radiation Biology*, vol. 70, pp. 665–669, 1996.

[73] R. Huber, U. Kulka, T. Lörch, *et al.*, « Technical report: application of the Metafer2 fluorescence scanning system for the analysis of radiation-induced chromosome aberrations measured by FISH-chromosome painting », *Mutation Research/Genetic Toxicology and Environmental Mutagenesis*, vol. 492, pp. 51–57, 2001.

[74] B. C. Shirley, J. H. M. Knoll, J. Moquet, *et al.*, « Estimating partial-body ionizing radiation exposure by automated cytogenetic biodosimetry », *International Journal of Radiation Biology*, vol. 96, *11*, pp. 1492–1503, Nov. 2020.

[75] A. Subasinghe, J. Samarabandu, Y. Li, *et al.*, « Centromere detection of human metaphase chromosome images using a candidate based method », F1000Research, Tech. Rep., 2016.

[76] P. Finnon, D. C. Lloyd, and A. A. Edwards, « An assessment of the metaphase finding capability of the Cytoscan 110 », *Mutation Research*, vol. 164, pp. 101–108, 1986.

[77] J. Piper and J. Sprey, « Adaptive classifiers for dicentric chromosomes », *Journal of Radiation Research*, vol. 33, pp. 159–170, 1992.

[78] P. Finnon and D. Lloyd, « A preliminary evaluation of the Edinburgh dicentric hunter », *Journal of Radiation Research*, vol. 33 Suppl, pp. 215–221, 1992.

[79] Y. Li, A. Wickramasinghe, A. A. Subasinghe, *et al.*, « Towards large scale automated interpretation of cytogenetic biodosimetry data », *in IEEE International Conference on Information and Automation for Sustainability*, 2012, pp. 30–35.

[80] N. Otsu, « A Threshold Selection Method from Gray-Level Histograms », *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, 1979.

[81] A. Subasinghe Arachchige, J. Samarabandu, J. Knoll, W. Khan, and P. Rogan, « An image processing algorithm for accurate extraction of the centerline from human metaphase chromosomes », *in IEEE International Conference on Image Processing*, 2010, pp. 3613–3616.

[82] A. S. Arachchige, J. Samarabandu, J. H. M. Knoll, and P. K. Rogan, « Intensity Integrated Laplacian-Based Thickness Measurement for Detecting Human Metaphase Chromosome Centromere Location », *IEEE Transactions on Biomedical Engineering*, vol. 60, pp. 2005–2013, 2013.

[83] X. Bai, L. Latecki, and W.-y. Liu, « Skeleton Pruning by Contour Partitioning with Discrete Curve Evolution », *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 449–462, 2007.

[84] J. Liu, Y. Li, R. Wilkins, F. Flegal, J. H. Knoll, and P. K. Rogan, « Accurate cytogenetic biodosimetry through automated dicentric chromosome curation and metaphase cell selection », *F1000Research*, vol. 6, p. 1396, 2017.

[85] Y. LI, J. Knoll, R. Wilkins, F. Norton, and P. Rogan, « Automated discrimination of dicentric and monocentric chromosomes by machine learning-based image processing », *Microscopy Research and Technique*, vol. 79, 2016.

[86] P. K. Rogan, Y. Li, A. Wickramasinghe, *et al.*, « Automating dicentric chromosome detection from cytogenetic biodosimetry data », *Radiation Protection Dosimetry*, vol. 159, pp. 95–104, 2014.

[87] A. S. A., J. Samarabandu, J. Knoll, W. Khan, and P. Rogan, « An Accurate Image Processing Algorithm for Detecting FISH Probe Locations Relative to Chromosome Landmarks on DAPI Stained Metaphase Chromosome Images », *in Canadian Conference on Computer and Robot Vision*, 2010, pp. 223–230.

[88] P. K. Rogan, Y. Li, R. C. Wilkins, F. N. Flegal, and J. H. M. Knoll, « Radiation Dose Estimation by Automated Cytogenetic Biodosimetry », *Radiation Protection Dosimetry*, vol. 172, pp. 207–217, 2016.

[89] Y. Li, B. C. Shirley, R. C. Wilkins, F. Norton, J. H. M. Knoll, and P. K. Rogan, « Radiation Dose Estimation By Completely Automated Interpretation Of The Dicentric Chromosome Assay », *Radiation Protection Dosimetry*, 2019.

[90] S. Jang, S.-G. Shin, M.-J. Lee, *et al.*, « Feasibility Study on Automatic Interpretation of Radiation Dose Using Deep Learning Technique for Dicentric Chromosome Assay », *Radiation Research*, vol. 195, pp. 163–172, 2021.

[91] X. Shen, T. Ma, C. Li, Z. Wen, J. Zheng, and Z. Zhou, « High-precision automatic identification method for dicentric chromosome images using two-stage convolutional neural network », *Scientific Reports*, vol. 13, p. 2124, Feb. 2023.

[92] P. Voisin, M. Benderitter, M. Claraz, *et al.*, « The cytogenetic dosimetry of recent accidental overexposure », *Cellular and Molecular Biology*, vol. 47, pp. 557–564, 2001.

[93] A. Vaurijoux, G. Gruel, F. Pouzoulet, *et al.*, « Strategy for population triage based on dicentric analysis », *Radiation Research*, vol. 171, pp. 541–548, 2009.

[94] A. Vaurijoux, E. Gregoire, S. Roch-Lefevre, *et al.*, « Detection of Partial-Body Exposure to Ionizing Radiation by the Automatic Detection of Dicentrics », *Radiation Research*, vol. 178, pp. 357–364, 2012.

[95] G. Gruel, E. Grégoire, S. Lecas, *et al.*, « Biological dosimetry by automated dicentric scoring in a simulated emergency », *Radiation research*, vol. 179, 2013.

[96] H. Romm, L. Ainsbury, S. Barnard, *et al.*, « Automatic scoring of dicentric chromosomes as a tool in large scale radiation accidents », *Mutation research*, vol. 756, May 2013.

[97] T. L. Ryan, M. B. Escalona, T. L. Smith, J. Albanese, C. J. Iddins, and A. S. Balajee, « Optimization and validation of automated dicentric chromosome analysis for radiological/nuclear triage applications », *Mutation Research/Genetic Toxicology and Environmental Mutagenesis*, vol. 847, p. 503 087, Nov. 2019.

[98] D. Endesfelder, U. Kulka, J. Einbeck, and U. Oestreicher, « Improving the accuracy of dose estimates from automatically scored dicentric chromosomes by accounting for chromosome number », *International Journal of Radiation Biology*, vol. 96, *12*, pp. 1571–1584, Dec. 2020.

[99] A. Defazio, F. Bach, and S. Lacoste-Julien, « SAGA: a fast incremental gradient method with support for non-strongly convex composite objectives », *in Advances in Neural Information Processing Systems (NeurIPS)*, 2014, pp. 1646–1654.

[100] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, « Scikit-learn: Machine Learning in Python », *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[101] R. Padilla, S. L. Netto, and E. A. B. da Silva, « A Survey on Performance Metrics for Object-Detection Algorithms », *in International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020, pp. 237–242.

[102] A. Bulat and G. Tzimiropoulos, « Binarized Convolutional Landmark Localizers for Human Pose Estimation and Face Alignment with Limited Resources », *in IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3726–3734.

[103] A. Bulat and G. Tzimiropoulos, « How far are we from solving the 2D and 3D Face Alignment problem? (and a dataset of 230,000 3D facial landmarks) », *in IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1021–1030.

[104] V. Lempitsky and A. Zisserman, « Learning To Count Objects in Images », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 23, 2010.

[105] J. P. Cohen, G. Boucher, C. A. Glastonbury, H. Z. Lo, and Y. Bengio, « Count-ception: Counting by Fully Convolutional Redundant Counting », 2017, pp. 18–26.

[106] P. Chattopadhyay, R. Vedantam, R. R. Selvaraju, D. Batra, and D. Parikh, « Counting Everyday Objects in Everyday Scenes », *in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4428–4437.

[107] I. H. Laradji, N. Rostamzadeh, P. O. Pinheiro, D. Vazquez, and M. Schmidt, « Where Are the Blobs: Counting by Localization with Point Supervision », *in European Conference on Computer Vision (ECCV)*, vol. 11206, 2018, pp. 560–576.

[108] F. Wang, G. Wei, Q. Liu, J. Ou, x. wei xian, and H. Lv, « Boost Neural Networks by Checkpoints », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021, pp. 19 719–19 729.

[109] R. E. Schapire, « A brief introduction to boosting », *in IJCAI International Joint Conference on Artificial Intelligence*, vol. 2, 1999, pp. 1401–1406.

[110] D. Marmanis, J. D. Wegner, S. Galliani, K. Schindler, M. Datcu, and U. Stilla, « Semantic Segmentation Of Aerial Images With An Ensemble Of CNNs », *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 3, pp. 473–480, 2016.

[111] K. Shaga Devan, H. A. Kestler, C. Read, and P. Walther, « Weighted average ensemble-based semantic segmentation in biological electron microscopy images », *Histochemistry and Cell Biology*, vol. 158, pp. 447–462, 2022.

[112] L. Nanni, C. Fantozzi, A. Loreggia, and A. Lumini, « Ensembles of Convolutional Neural Networks and Transformers for Polyp Segmentation », *Sensors*, vol. 23, p. 4688, 2023.

[113] A. Neubeck and L. Van Gool, « Efficient Non-Maximum Suppression », *in International Conference on Pattern Recognition (ICPR)*, 2006, pp. 850–855.

[114] R. Solovyev, W. Wang, and T. Gabruseva, « Weighted boxes fusion: Ensembling boxes from different object detection models », *Image and Vision Computing*, vol. 107, p. 104 117, 2021.

[115] A. G. Wilson and P. Izmailov, « Bayesian Deep Learning and a Probabilistic Perspective of Generalization », *in Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 4697–4708.

[116] G. Franchi, A. Bursuc, E. Aldea, S. Dubuisson, and I. Bloch, « TRADI: Tracking Deep Neural Network Weight Distributions », *in European Conference on Computer Vision (ECCV)*, Series Title: Lecture Notes in Computer Science, vol. 12362, 2020, pp. 105–121. DOI: 10.1007/978-3-030-58520-4_7.

[117] M. Li and C. Scheidegger, *Comparing Deep Neural Nets with UMAP Tour*, 2021.

[118] L. McInnes, J. Healy, N. Saul, and L. Großberger, « UMAP: Uniform Manifold Approximation and Projection », *Journal of Open Source Software*, vol. 3, *29*, p. 861, 2018.

[119] J. C. Gower, « Generalized procrustes analysis », *Psychometrika*, vol. 40, pp. 33–51, 1975.

[120] A. Araujo, W. Norris, and J. Sim, « Computing Receptive Fields of Convolutional Neural Networks », *Distill*, 2019.

[121] M. Pachitariu and C. Stringer, « Cellpose 2.0: how to train your own model », *Nature Methods*, vol. 19, *12*, pp. 1634–1641, Dec. 2022, Number: 12 Publisher: Nature Publishing Group.

[122] W. Zhang, J. Pang, K. Chen, and C. C. Loy, « Dense Siamese Network for Dense Unsupervised Learning », *in European Conference on Computer Vision (ECCV)*, 2022, pp. 464–480.

[123] J. Kun, M. Smieja, B. Xiong, L. Soleymani, and Q. Fang, « The Use of Motion Analysis as Particle Biomarkers in Lensless Optofluidic Projection Imaging for Point of Care Urine Analysis », *Scientific Reports*, vol. 9, p. 17 255, 2019.

[124] B. K. P. Horn and B. G. Schunck, « Determining optical flow », *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[125] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, « High Accuracy Optical Flow Estimation Based on a Theory for Warping », *in European Conference on Computer Vision (ECCV)*, T. Pajdla and J. Matas, Eds., 2004, pp. 25–36.

[126] E. Mémin and P. Pérez, « Hierarchical Estimation and Segmentation of Dense Motion Fields », *International Journal of Computer Vision*, vol. 46, pp. 129–155, 2002.

[127] D. Fortun, P. Bouthemy, and C. Kervrann, « Optical Flow Modeling and Computation: A Survey », *Computer Vision and Image Understanding*, vol. 134, 2015.

[128] B. D. Lucas and T. Kanade, « An iterative image registration technique with an application to stereo vision », *in International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 2, 1981, pp. 674–679.

[129] E. Mémin and P. Pérez, « Dense estimation and object-based segmentation of the optical flow with robust techniques », *IEEE Transactions on Image Processing*, vol. 7, pp. 703–719, 1998.

[130] J. Delpiano, J. Jara, J. Scheer, O. A. Ramírez, J. Ruiz-del-Solar, and S. Härtel, « Performance of optical flow techniques for motion analysis of fluorescent point signals in confocal microscopy », *Machine Vision and Applications*, vol. 23, pp. 675–689, 2012.

[131] D. Fortun, P. Bouthemy, P. Paul-Gilloteaux, and C. Kervrann, « Aggregation of patch-based estimations for illumination-invariant optical flow in live cell imaging », *in IEEE International Symposium on Biomedical Imaging*, 2013, pp. 660–663.

[132] D. Fortun, N. Debroux, and C. Kervrann, « Spatially-Variant Kernel for Optical Flow Under Low Signal-to-Noise Ratios Application to Microscopy », *in IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 42–48.

[133] S. Manandhar, P. Bouthemy, E. Welf, G. Danuser, P. Roudot, and C. Kervrann, « 3D flow field estimation and assessment for live cell fluorescence microscopy », *Bioinformatics*, vol. 36, *5*, pp. 1317–1325, 2020.

[134] D. Fortun, P. Bouthemy, and C. Kervrann, « Aggregation of local parametric candidates with exemplar-based occlusion handling for optical flow », *Computer Vision and Image Understanding*, Light Field for Computer Vision, vol. 145, pp. 81–94, 2016.

[135] A. Dosovitskiy, P. Fischer, E. Ilg, *et al.*, « FlowNet: Learning Optical Flow with Convolutional Networks », *in IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2758–2766.

[136] Z. Teed and J. Deng, « RAFT: Recurrent All-Pairs Field Transforms for Optical Flow », *in European Conference on Computer Vision (ECCV)*, vol. 12347, 2020, pp. 402–419.

[137] R. Jonschkowski, A. Stone, J. T. Barron, A. Gordon, K. Konolige, and A. Angelova, « What Matters in Unsupervised Optical Flow », *in European Conference on Computer Vision (ECCV)*, 2020, pp. 557–572.

[138] L. Liu, J. Zhang, R. He, *et al.*, « Learning by Analogy: Reliable Supervision From Transformations for Unsupervised Optical Flow Estimation », *in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6488–6497.

[139] Z. Huang, X. Shi, C. Zhang, *et al.*, « FlowFormer: A Transformer Architecture for Optical Flow », *in European Conference on Computer Vision (ECCV)*, vol. 13677, 2022, pp. 668–685.

[140] A. Stone, D. Maurer, A. Ayvaci, A. Angelova, and R. Jonschkowski, « SMURF: Self-Teaching Multi-Frame Unsupervised RAFT with Full-Image Warping », *in IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 3886–3895.

[141] S. Berg, D. Kutra, T. Kroeger, *et al.*, « Ilastik: interactive machine learning for (bio)image analysis », *Nature Methods*, vol. 16, pp. 1226–1232, 2019.

COLLEGE DOCTORAL BRETAGNE
MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

Université de Rennes

Titre : titre (Apprentissage machine et réseaux de convolution pour une expertise augmentée en dosimétrie biologique

Mot clés : Apprentissage profond, Agrégation de modèles, Incertitude, Détection d'objets, Imagerie médicale

Résumé : La dosimétrie biologique est la branche de la physique de la santé qui se préoccupe de l'estimation de doses de rayonnement ionisants à partir de biomarqueurs. Dans le procédé standard défini par l'AIEA, la dose est calculée en estimant la fréquence d'apparition de chromosomes dicentriques lors de la métaphase des lymphocytes périphériques. La variabilité morphologique des chromosomes, ainsi que celle des conditions d'acquisition des images rend ce problème de détection d'objets complexe. De plus, l'estimation fiable de cette fréquence nécessite le traitement d'un grand nombre d'image. Etant donné les limites du comptage humain (faible nombre de personnes qualifiées, charge cognitive), l'automatisation est une nécessité dans le contexte d'exposition de masse. Dans ce contexte, l'objectif de cette thèse est de tirer parti des progrès récents en vision par ordinateur (et plus spécifiquement en détection d'objets) apportés par l'apprentissage profond. La contribution principale de ce travail est une preuve de concept pour un modèle de détection de chromosomes dicentriques. Ce système repose sur l'agrégation de modèles pour parvenir à un haut niveau de performance, ainsi qu'a une bonne quantification de son incertitude, une exigence essentielle dans un contexte médical.

Title: titre (Machine learning and convolutional networks for automated biological dosimetry

Keywords: Deep learning, Model Agregation, Uncertainty, Object detection, Medical imaging

Abstract: Biological dosimetry is the branch of health physics dealing with the estimation of ionizing radiation doses from biomarkers. The current gold standard (defined by the IAEA) relies on estimating how frequently dicentric chromosomes appear in peripheral blood lymphocytes. Variations in acquisition conditions and chromosome morphology makes this a challenging object detection problem. Furthermore, the need for an accurate estimation of the average number of dicentric per cell means that a large number of image has to be processed. Human counting is intrinsically limited, as cognitive load is high and the number of specialist insufficient in the context of a large-scale exposition. The main goal of this PhD is to use recent developments in computer vision brought by deep learning, especially for object detection. The main contribution of this thesis is a proof of concept for a dicentric chromosome detection model. This model agregates several Unet models to reach a high level of performance and quantify its prediction uncertainty, which is a stringent requirement in a medical setting.