Prédiction d'efficience d'algorithme de placement

Corentin Lallier, Bruno Pinaud, Marc Palyart, Laurent Vézard, Guillaume Blin

Apprentissage profond : prédiction d'efficience de placement pour la découpe industrielle de textiles

Corentin Lallier*,**, Bruno Pinaud*, Marc Palyart**, Laurent Vézard**, Guillaume Blin*

*LaBRI UMR CNRS 5800 – Université de Bordeaux, Bât A30,
351 cours de la Libération, 33405 Talence Cedex
{prénom}.{nom}@u-bordeaux.fr,
https://www.labri.fr
**Lectra - 23 Chemin de Marticot, 33610 Cestas
{initiale prénom}.{nom}@lectra.com
https://www.lectra.com

Résumé. Dans un processus industriel de découpe de matières textiles, les minimisations des pertes de matières ainsi que des temps de coupe sont des préoccupations majeures. Le temps peut être optimisé en découpant plusieurs couches de matières en même temps. Optimiser la matière est en revanche un problème complexe. Afin d'être découpées, les pièces sont regroupées par lots. Obtenir les regroupements les plus efficients en matière nécessite le calcul de la consommation de matière pour chaque lot possible. L'efficience matière d'un lot est obtenue grâce à un algorithme de placement qui positionne les pièces sur la matière. Cet algorithme est très coûteux en temps et financièrement, rendant impossible le calcul de l'ensemble des lots. Actuellement, le problème est résolu à l'aide de tables de références approximatives conçues empiriquement qui donnent le taux d'utilisation matière pour chaque lot. Dans cet article, nous présenterons les travaux de ma première année de thèse : une première approche en apprentissage profond permettant d'estimer l'efficience du placement d'un lot à partir d'une base de données construites sur des placements existants. Nous proposons une combinaison de techniques de modélisation des pièces à l'aide de réseaux de neurones convolutionels et de modélisation d'un lot de pièces grâce à des méthodes de type *Multiple Instance Learning*. Nous présentons aussi les résultats préliminaires liés à cette approche et nos plans pour la suite.

1 Introduction

L'utilisation de textile est prépondérante dans de nombreux domaines tels que la mode (vêtements, accessoires, chaussures), l'automobile (sièges, intérieurs, airbags) ou encore l'ameublement (canapés, fauteuils). La minimisation des pertes de matières textiles lors de la fabrication industrielle de ces articles est une préoccupation majeure ;

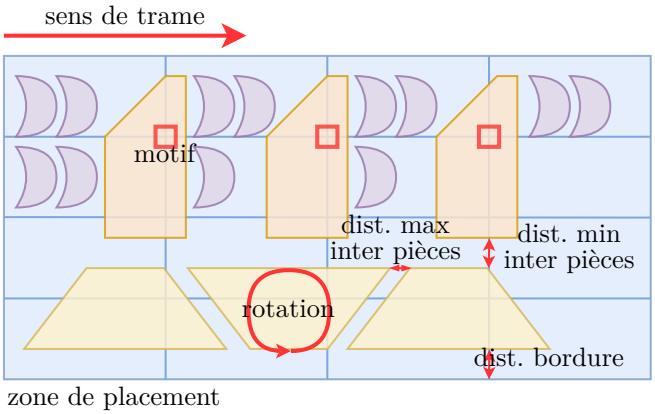

FIG. 1 – *Schéma d'un placement. La zone bleue représente la surface rectangulaire du matériau à découper (aussi appelée zone de placement). Les polygones sont les pièces à découper. Les contraintes sont représentées en rouge.*

autrefois pour des raisons purement économiques et plus récemment aussi pour des raisons environnementales (Rissanen et McQuillan, 2016; Henninger et al., 2016).

Au-delà de la conception elle-même du produit (forme et composition), deux facteurs permettent de réduire les chutes de textile : la façon dont les pièces des articles sont regroupées par lot (couramment appelé plan de sections) pour être découpées sur des rouleaux ou des matelas de matériaux (empilement de couches de tissus) et la façon dont ces pièces sont disposées sur la matière pour la découpe (appelé placement). La réalisation d'un plan de sections nécessite de connaître l'efficience (proportion de matière réellement utilisée) de chaque groupe possible pour déterminer les groupements les plus efficients. Malheureusement, la réalisation d'un placement pour chaque combinaison de regroupement possible n'est pas envisageable car l'opération de placement est trop coûteuse tant en temps de calcul que financièrement. À la place on se base sur des tables de références approximatives conçues empiriquement via la réalisation de quelques placements types. Par exemple, on va décider qu'un placement mélangeant des pantalons basiques de taille S et M va aboutir à une efficience de 80%. Cette approximation conduit souvent à ne pas faire les regroupements optimaux. Dans cette optique, nos travaux visent à développer un algorithme capable de prédire rapidement l'efficience attendue d'un placement sans le réaliser afin d'améliorer la performance de la construction des plans de sections.

Le plan de cet article est le suivant : la partie 2 présente le problème du placement plus en détails, puis dans la partie 3 nous présentons notre approche de la modélisation du problème. Ensuite, la partie 4 détaille l'architecture du modèle qui répond à cette modélisation et en particulier les premiers résultats dans la partie 4.2. Enfin, nous concluons (partie 5) sur nos travaux futurs.

2 Problématique

Un placement consiste à positionner convenablement les pièces sur le rouleau ou matelas de matériau à découper en respectant les contraintes imposées, tout en minimisant la perte de matière.

La figure 1 présente les principaux éléments d'un placement. La zone de placement, ici la zone bleue, représente la surface rectangulaire du matériau à découper, par exemple un matelas de tissu. Les pièces représentent les formes issues du patron CAO (conception assistée par ordinateur). Ces formes sont représentées par des polygones. Par exemple, dans un patron de chemise, une manche est représentée par un trapèze (pièce plus large à l'épaule qu'au niveau des poignets).

L'algorithme de placement doit répondre à un certain nombre de contraintes liées aux pièces ou à la zone de placement. Sur la figure 1, on peut voir certains exemples de contraintes en rouge. Généralement les contraintes sont liées à la machine de découpe ou à la matière à découper. Si la matière à découper est un tissu, alors les pièces vont devoir respecter la trame du tissu (contrainte **sens de trame**) afin de répondre à des critères de résistance et de qualité de découpe et seules certaines rotations respectant la trame sont autorisées (contraintes **rotation**).

De plus, dans les rouleaux de tissus à découper, les bordures sont généralement à éviter, car moins résistantes et les colorations ne sont pas uniformes sur toute la longueur du rouleau. Sur ce type de matériau, les pièces proches dans l'article final, par exemple deux parties d'une manche d'une chemise, vont devoir être placées sur des zones de couleur ou de motifs similaires. Ceci est modélisé par les contraintes de **distances maximum entre pièces**, de **distance à la bordure** et de **motifs**.

De même, si deux pièces sont trop proches, le découpage d'une pièce peut dégrader la pièce adjacente du fait de contraintes physiques de la machine de découpe comme la largeur de la lame. Cette contrainte est nommée **distance minimale entre pièces**.

L'efficience est donnée par le rapport entre la somme des surface des pièces sur la surface de la zone de placement. Notre problématique générale est de pouvoir **estimer rapidement** l'efficience d'un placement, donnée par un algorithme de placement, **sans exécuter cet algorithme**, en fonction de la laize, des pièces à y placer et des contraintes qui s'appliquent.

3 Modélisation de l'estimation de l'efficience

Pour estimer l'efficience d'un placement sans le réaliser, nous utilisons une base d'apprentissage de placements sur laquelle nous appliquons des techniques de régression en apprentissage automatique. Nous décomposons le problème d'estimation d'efficience ainsi : 1) modélisation des pièces, 2) modélisation d'ensembles de pièces, et 3) modélisation des relations et contraintes entre pièces.

Cet article traite des deux premiers points. Le troisième est abordé dans la discussion finale (partie 5) pour nos futurs travaux.

3.1 Modélisation haut-niveau

Pour répondre à ces deux premiers points, nous proposons l'architecture présentée dans la figure 2. Elle se base sur un modèle multi-entrées et combine plusieurs modules spécifiques. Le premier module, illustré en figure 2 (a), est dédié à la modélisation des pièces. Il est spécialisé dans la reconnaissance de formes et repose sur une succession de convolutions. Son rôle est d'extraire les caractéristiques saillantes de chaque image, afin

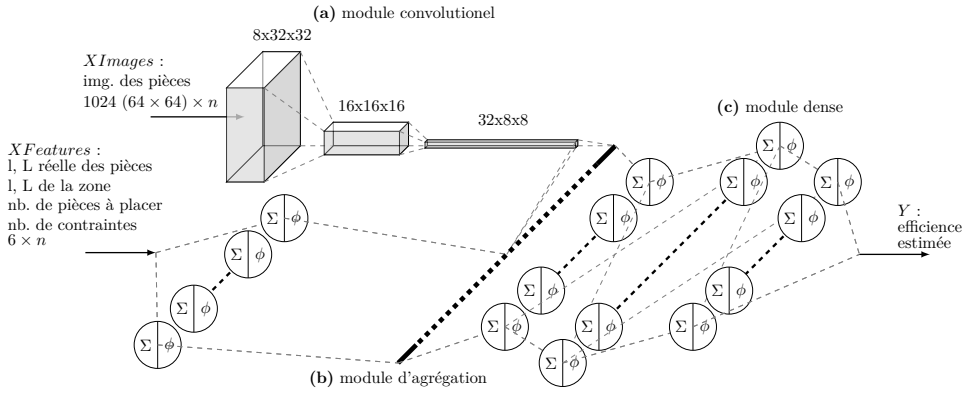

FIG. 2 – *Schéma du modèle. (a) : module de convolutions, dédié à la modélisation des pièces. (b) : module d'agrégation MIL Pool, dédié à la modélisation d'ensembles de pièces. (c) : module dense, dédié à la décision. Ici n est le nombre de pièces d'un placement.*

d'en créer une représentation simplifiée, sous la forme d'un vecteur basse dimension. Il prend en entrée, pour chaque placement, la liste des images rasterisées des pièces du placement (notées $XImages$) et fournit en sortie un vecteur pour chaque image appelé *embeddings* (voir figure 3). Par la suite, nous y ferons référence comme "module de convolutions".

Le second module, illustré en figure 2 (*b*), est dédié à la modélisation d'ensembles de pièces. Son rôle est de créer un vecteur unique à faible dimension à partir des listes de vecteurs fournies par le module précédent. Il prend en entrée les représentations construites par le module de convolutions, ainsi que les vecteurs de caractéristiques associés à chaque image des pièces (notées $XFeatures$) et fournit en sortie un vecteur pour chaque placement. Par la suite, nous y ferons référence comme "module d'agrégation".

Le dernier module est un module dense, illustré en figure 2 (*c*), dédié à la décision. Il prend en entrée le vecteur de sortie du module d'agrégation, représentant le placement et effectue la régression vers une valeur scalaire. Cette valeur représente l'efficience prédite du placement. La fonction de coût est calculée sur cette valeur. Par la suite, nous y ferons référence comme "module dense".

3.2 Modélisation des pièces

La reconnaissance de formes est généralement effectuée par des réseaux de neurones convolutionels (CNNs, LeCun et al. (1999)). Ce sont des réseaux hiérarchiques à multiples couches. Chaque couche transforme les données d'entrée via un ensemble de noyaux de convolutions, appelés filtres, qui vont être optimisés pendant l'apprentissage. Ces réseaux se basent sur les concepts de **champs réceptifs locaux**, de **partage des poids** et de **sous-échantillonage spatial** pour dégager une invariance spatiale partielle entre les pièces. Les champs réceptifs locaux sont définis par le fait que chaque

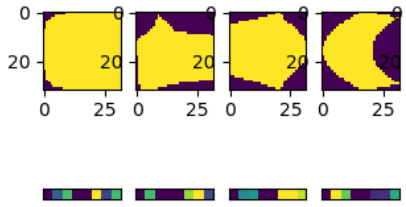

FIG. 3 – *Exemple de données du module de convolutions de la Figure 2-(a). En haut, les images correspondant aux formes des pièces rasterisées, en entrée du module. En bas, les embeddings de sortie, générés par le module.*

unité, ou neurone, d'une couche prend en entrée les valeurs de sortie des unités de la couche précédente situées dans son voisinage. Cette opération a pour but d'extraire des descripteurs visuels élémentaires locaux tels que des bordures ou des coins. Ces descripteurs sont ensuite combinés dans les couches plus profondes pour répondre à des caractéristiques de plus haut niveau (combinaisons de bordures et de coins). Cette idée d'invariance spatiale est inspirée par la découverte de neurones réagissant localement à des stimulus visuels précis dans le cortex visuel du chat (Hubel et Wiesel, 1959). Les unités d'une couche partagent le même ensemble de poids. Elles sont ainsi contraintes à apprendre et appliquer les mêmes filtres par couche. Le sous-échantillonage (ou pooling) est une méthode pour réduire la résolution spatiale des représentations intermédiaires (appelées cartes de caractéristiques). À chaque couche du réseau, les unités appliquent une opération d'agrégation sur les entrées de leur champ réceptif pour réduire la résolution spatiale. Cela permet, de plus, de construire des relations spatiales entre les descripteurs comme le modèle prototypique de ces architectures, baptisé LeNet (LeCun et al., 1999; Lecun et al., 1998).

L'architecture moderne des CNN a été proposée avec Alexnet (Krizhevsky et al., 2017) sous la forme d'une alternance de couches de convolutions et de couches d'agrégation *maximum*, appelées max-pooling. Via un balayage par fenêtrage des unités de la carte de caractéristiques, cet opérateur considère le maximum des valeurs du voisinage de chaque unité comme nouvelle représentation. Ce fenêtrage est appliqué avec un pas pour réduire la taille de la carte de caractéristiques et ainsi compresser l'information. Chacune de ces couches est suivie d'une fonction d'activation, dont le but est de filtrer la sortie de chaque unité, généralement en fonction d'un seuil. Cette architecture est, elle aussi, suivie d'un petit nombre de couches denses, dans lesquelles chaque unité est connectée aux unités de la couche précédente, sans notion de voisinage (Springenberg et al., 2015; Khan et al., 2020).

Nous allons maintenant présenter notre approche de modélisation des ensembles de tailles variables de ces pièces.

3.3 Modélisation d'ensembles de pièces

Plusieurs familles de méthodes classiques sont utilisables pour modéliser des ensembles, par exemple, les LSTM -*Long Short-Term Memory* (Hochreiter et Schmidhuber, 1997) ou les *GRU Gated Recurrent Units* (Cho et al., 2014). Ces méthodes sont issues de champs de recherches particuliers (traitement du langage, séries temporelles) et nécessitent généralement que les données soient ordonnées.

Dans cet article, nous nous intéressons aux méthodes du type *Multiple Instance Learning* ou MIL qui sont des formes d'apprentissages semi-supervisés. Nous pensons que ces méthodes sont plus adaptées aux données d'ensembles de pièces, qui n'ont pas de notion d'ordre. L'hypothèse standard en MIL est qu'un label est fourni pour un ensemble d'instances. Chacun de ces ensembles est appelé *bag*. Cette méthode est semi-supervisée, car les labels ne sont connus qu'au niveau des bags et seulement partiellement au niveau des instances (les données d'entrées). Un bag est considéré comme positif si a minima une de ses instances est positive. Un bag est considéré comme négatif si l'ensemble des instances qu'il contient sont négatives (Carbonneau et al., 2018).

Nous considérons le MIL comme une approche permettant de comparer des ensembles de tailles variables non ordonnés : le principe est d'agréger un ensemble d'instances (un ensemble de vecteurs) en un unique vecteur. Le label est associé à cette représentation agrégée pour permettre une classification ou une régression.

Dans les réseaux de neurones, on considère que les instances sont des caractéristiques extraites de couches de réseaux. On peut les voir comme des représentations basse dimension des vecteurs d'entrées, sur lesquelles on applique une agrégation de type MIL, appelé MIL Pool. Cette opération d'agrégation doit être invariante à l'ordre dans lequel les instances sont présentées (Ilse et al., 2018).

4 Architecture

Notre modèle est basé sur une architecture classique en apprentissage profond, dans laquelle on peut voir une partie extraction de caractéristiques suivie d'une partie de compression par agrégation (pooling), et enfin une partie décision (Lecun et al., 1998).

En se basant sur les architectures des CNN présentées précédemment, nous avons choisi pour le module de convolution de s'inspirer du modèle LeNet en y apportant les fonctions d'activations proposées par Alexnet et en remplaçant les couches de sous-échantillonnages par un pas de fenêtrage (ou *stride*) ayant une plus basse complexité computationelle. Le module de convolution est composé de trois couches de convolutions, utilisant chacune une fonction d'activation de type ReLU (Rectified Linear Unit). Chacune de ces couches a une taille de voisinage de deux et un pas de fenêtrage (ou stride) de deux permettant la réduction d'informations à chaque couche.

4.1 Implémentation

L'implémentation, basée sur Pytorch (Paszke et al., 2019), prend en entrée des lots de placements où chaque placement est décrit à l'aide de $XImages$ et de $XFeatures$. Les $XImages$ sont des images rasterisées des pièces du placement en résolution 64×64 pixels. Les $XFeatures$ sont des caractéristiques de ces pièces : largeur et longueur réelle de chaque pièce, taille de la zone de placement, nombre de pièces à placer et nombre de contraintes. Les dimensions pour $XImages$ sont de $1024 \times n$ et pour $XFeatures$ sont de $6 \times n$. Avec n le nombre de pièces du placement.

Les tailles des pièces et la taille de la zone du placement sont fournies avec une précision de l'ordre du dix-millième de mètre.

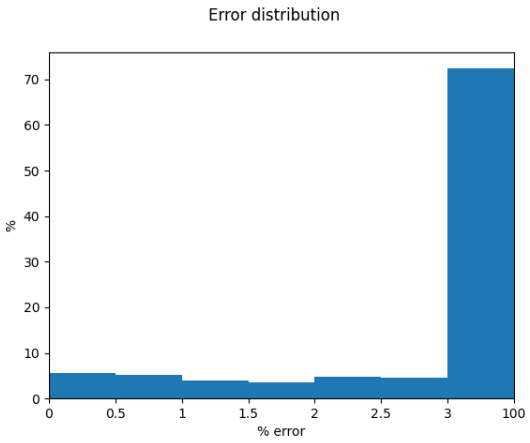

FIG. 4 – *Distribution de l'erreur absolue du modèle, par pas de 0.5% en dessous de 3% puis au-delà de 3%*

Une étude antérieure relative à l'identification de pièces de patrons montre qu'il y a un compromis à faire pour les dimensions des images des pièces utilisées en entrée entre la qualité et les temps d'entraînement : plus ces images sont grandes, plus la qualité de la reconnaissance augmente, ainsi que la complexité, le temps et l'espace des calculs. Au-delà d'une résolution de 64×64 pixels, la reconnaissance de pièces ne bénéficie plus de l'apport d'informations fournies (Jouanneau, 2019). La sortie du module de convolution, appelée embedding, est un vecteur à 16 dimensions.

4.2 Résultats préliminaires

La base de données utilisée comporte 100000 placements. Nous entraînons notre modèle sur 90000 placements dont 10% servent (soit 9000) à la validation à chaque fin de cycle d'apprentissage (ou *epochs*). Les 10000 placements restants sont utilisés pour la phase d'évaluation. L'apprentissage est fait sur cinq epochs, car l'amélioration des résultats de l'étape de validation stagnait. Nous utilisons Adam *Adaptative Moment Estimation* (Kingma et Ba, 2014) comme méthode d'optimisation de descente de gradient avec un taux d'apprentissage de 0.01 et la distance $L1$ (moyenne des différences absolues entre les prédictions et les efficiences réelles) comme fonction de coûts.

Pour l'évaluation, nous utilisons trois métriques courantes (Botchkarev, 2019) et une métrique métier de distribution des erreurs. Le modèle produit une erreur quadratique moyenne (Mean Squared Error, MSE) faible (1.7%), mais une erreur moyenne (Mean Absolute Error, MAE) élevée ($7,75\%$ d'erreur). L'erreur maximale (soit 82.6% dans le pire cas) est simplement inacceptable. La distribution de l'erreur absolue en fonction des placements évalués est représentée dans la figure 4. Cet histogramme montre la répartition d'erreur par pas de 0.5% entre 0 et 3% d'erreur, puis au-delà de 3%. Cette répartition est utilisée comme métrique de comparaison métier. Notre but est d'avoir une erreur moyenne en dessous de 3% et idéalement autour de 1% pour être utilisable en pratique dans l'optimisation des plans de sections.

5 Conclusion et travaux futurs

Cet article est une première approche de modélisation et d'implémentation d'un problème d'estimation d'efficience de processus de découpe industrielle de matières textiles. Ces premiers résultats sont plutôt décevants, mais ouvre la voie a de futures expérimentations.

Tel que présenté, l'erreur moyenne des prédictions du modèle est de $7,8\%$, notre cible étant en dessous de 3% pour l'optimisation des plans de sections. Nous pensons qu'un des problèmes principaux est lié à la distribution des exemples d'apprentissage. En effet, notre base d'apprentissage est créée à partir de données réelles et est donc biaisée, les placements globalement efficients étant sur-représentés par rapport aux autres. La distribution des efficiences des placements a une moyenne supérieure à 70% et un écart-type faible (environ 10). Une solution serait de ré-échantillonner de manière plus homogène notre base d'apprentissage pour être plus représentatif des cas plus rares en particulier une faible efficience ($< 70\%$).

Un autre point d'amélioration serait la prise en compte des contraintes, comme présenté dans la partie 3. Elles ne sont pas utilisées actuellement. On peut voir le problème de la représentation des contraintes comme un problème de représentation de graphe, dans lequel les contraintes sont des relations entre les pièces à placer. Dans cette optique, les ensembles formes-contraintes sont représentées par un graphe dans lequel les pièces sont les sommets et les contraintes sont les arêtes du graphe. Prédire une efficience donnée revient à un problème de comparaison de graphes (par exemple en classification ou en régression). Le champ de recherche lié à ce domaine est celui de l'apprentissage géométrique. Les modèles d'apprentissage profond sont performants dans les traitements du langage, des images ou des vidéos pour lesquels les données ont une structure euclidienne. Récemment, une nouvelle famille de techniques est apparue, appelée apprentissage géométrique profond (Bronstein et al., 2017) et essayant de généraliser l'application des modèles d'apprentissages profonds aux données non-euclidiennes comme les graphes (ex : réseaux sociaux, réseaux génétiques, conception de molécules (Duvenaud et al., 2015), systèmes de recommandation) ou des géométries à plus de deux dimensions (modèles de déformations 3D, simulations physiques 3D (Sanchez-Gonzalez et al., 2020)) Deux approches principales sont utilisées pour ces modèles, soit de créer une représentation ou *embedding* pour chaque sommet (ex : Graph Convolution Network, GraphSage, Gated Graph Neural Networks), soit de créer des représentations de sous-graphes. Nous proposons d'explorer ces méthodes pour nos futurs travaux.

Le troisième point concerne les variations autour du modèle actuel. Particulièrement autour de la partie MIL Pool. Le MIL est un champ de recherche large et très documenté. Son application aux réseaux de neurones est assez récente et certains opérateur d'agrégation pourraient améliorer les performances de notre modèle, par exemple l'approche attentionelle présentée dans Ilse et al. (2018). Une autre approche intéressante serait de décomposer notre modèle en sous-modèles puis de les pré-entraîner pour une tâche spécifique, comme la reconnaissance de forme, puis de recombiner ces modèles, à la manière du *transfer learning*.

Références

Botchkarev, A. (2019). Performance metrics (error measures) in machine learning regression, forecasting and prognostics : Properties and typology. *Interdisciplinary Journal of Information, Knowledge, and Management 14*, 045–076.

Bronstein, M. M., J. Bruna, Y. LeCun, A. Szlam, et P. Vandergheynst (2017). Geometric deep learning : going beyond euclidean data. *IEEE Signal Processing Magazine 34*(4), 18–42.

Carbonneau, M.-A., V. Cheplygina, E. Granger, et G. Gagnon (2018). Multiple instance learning : A survey of problem characteristics and applications. *Pattern Recognition 77*, 329–353.

Cho, K., B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, et Y. Bengio (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, pp. 1724–1734. Association for Computational Linguistics.

Duvenaud, D., D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, et R. P. Adams (2015). Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, et R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Volume 28, pp. 2224–2232. Curran Associates, Inc.

Henninger, C. E., P. J. Alevizou, et C. J. Oates (2016). What is sustainable fashion ? *Journal of Fashion Marketing and Management : An International Journal 20*(4), 400–416.

Hochreiter, S. et J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 1735–1780.

Hubel, D. H. et T. N. Wiesel (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology 148*(3), 574–591.

Ilse, M., J. Tomczak, et M. Welling (2018). Attention-based deep multiple instance learning. In J. Dy et A. Krause (Eds.), *Proceedings of Machine Learning Research*, Volume 80, pp. 2127–2136. PMLR.

Jouanneau, W. (2019). Identification de patrons de vêtements et de pièces par machine learning. Rapport confidentiel interne, EINSEIRB-MATMECA / LECTRA.

Khan, A., A. Sohail, U. Zahoora, et A. S. Qureshi (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review 53*, 5455–5516.

Kingma, D. et J. Ba (2014). Adam : A method for stochastic optimization. In *International Conference on Learning Representations*.

Krizhevsky, A., I. Sutskever, et G. E. Hinton (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM 60*(6), 84–90.

Lecun, Y., L. Bottou, Y. Bengio, et P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*(11), 2278–2324. Conference

Name : Proceedings of the IEEE.

LeCun, Y., P. Haffner, L. Bottou, et Y. Bengio (1999). Object recognition with gradient-based learning. In D. A. Forsyth, J. L. Mundy, V. di Gesú, et R. Cipolla (Eds.), *Shape, Contour and Grouping in Computer Vision*, Lecture Notes in Computer Science, pp. 319–345. Springer.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, et S. Chintala (2019). PyTorch : An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, et R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32*, pp. 8026–8037. Curran Associates, Inc.

Rissanen, T. et H. McQuillan (2016). *Zero waste fashion design*, Volume 57. Bloomsbury Publishing.

Sanchez-Gonzalez, A., J. Godwin, T. Pfaff, R. Ying, J. Leskovec, et P. W. Battaglia (2020). Learning to simulate complex physics with graph networks. In *Thirty-seventh International Conference on Machine Learning (ICML)*.

Springenberg, J., A. Dosovitskiy, T. Brox, et M. Riedmiller (2015). Striving for simplicity : The all convolutional net. In *International Conference on Learning Representations (ICLR, workshop track)*.

Summary

In an industrial process of textile materials cutting, the minimization of both material losses and cutting times are major concerns. Time can be minimized by cutting a stack of material. However, optimizing the material is a complex problem. In order to be cut, parts have to be grouped into sets. Obtaining the most efficient material groupings requires the calculation of material consumption for each possible set. Efficiency is given by a nesting algorithm that positions parts on the material. This algorithm is expensive in time and money, making it impossible to calculate all the possible sets. Currently, the problem is solved using empirically designed reference tables which approximate the material utilization rate for each set.

In this paper, we present the work of my first year of PHD: a first deep learning approach to estimate the efficiency of the nesting of a set from a database built on existing nestings. We propose a combination of techniques from modeling the parts using convolutional neural networks and modeling a set of parts using *Multiple Instance Learning*. We also present the preliminary results related to this approach and our futur work.