Convolutional Neural Network programs meta-modeling for automatic detection of design smells

Aurel Ikama[1], Foutse Khomh[1], Heng Li[1]
[1] *Dep. of Computer and Software Engineering, Polytechnique Montréal, Montréal*

***Abstract*—TODO: To writte after the paper is finished**
***Keywords*—** Deep Learning, Convolutional Neural Network, Design Smells, Meta-Modeling, Software Engineering

I. Introduction

L'apprentissage profond est un sous ensemble du machine learning qui utilise des réseaux de neurones artificiels pour apprendre des données non structurées. L'apprentissage profond est devenu très populaire ces dernières années grâce à ses performances dans plusieurs domaines comme la reconnaissance d'image, la reconnaissance vocale, la traduction automatique, etc. Les réseaux de neurones artificiels sont des modèles mathématiques qui sont inspirés du fonctionnement du cerveau humain. Ils sont composés de plusieurs couches de neurones qui sont connectés entre eux. Chaque neurone est composé d'une fonction d'activation qui permet de calculer la sortie du neurone en fonction de ses entrées. Il existe plusieurs types d'architecture de réseaux de neurones artificiels. Ses architecture ont été développées pour répondre à des problèmes spécifiques tels que l'architecture de convolution pour la reconnaissance d'image, les réseaux de neurones récurrents pour la reconnaissance vocale, etc. Dans ce papier, nous nous intéressons à l'architecture de convolution car elle permet de traiter un large ensemble de problèmes comme la reconnaissance d'image, la reconnaissance de texte, la reconnaissance de séquence, etc. Par ailleurs l'architecture de convolution fait partie des architectures feed-forward qui sont une classe de réseaux de neurones artificiels qui sont composés de plusieurs couches de neurones et qui ne contiennent pas de cycles. Les réseaux de neurones feed-forward sont les plus utilisés dans l'apprentissage profond.

Nous parlerons dans ce papier de programmes d'apprentissage profond pour désigner les programmes contenant les réseaux de neurones. Tout comme n'importe quel programme, les programmes d'apprentissage profond peuvent contenir des défauts. Nous nous intéressons dans ce papier aux défauts de conception car se sont des défauts introduits tôt dans le cycle de développement du logiciel et ils peuvent avoir un impact négatif conséquent sur la performance et la qualité du logiciel. En effet les défauts de conception sont des défauts qui sont introduits par le développeur lors de la phase de conception du logiciel.

- Définition des défauts de conception
- Définit les défauts de conception dans les programmes d'apprentissage profond
- Définition différence entre odeurs, bugs, erreurs et défauts

Méta modélisation des programmes d'apprentissage profond

RQ1: Comment détecter les défauts de conception dans les programmes d'apprentissage profond?

RQ2: Quels sont les défauts de conception les plus répandu dans les programmes d'apprentissage profond CNN?

RQ3: Exist-il des lien entre les défauts de conception dans les programmes?

II. Study Design

In this sub-section, we describe the details of the data collection and processing approach followed to answer our different research questions.

A. Data Collection

Le développement du système de détection d'odeurs de conception s'est fait à partir des exemples de code de programmes d'apprentissage profond. Ces exemples représentent des programmes d'apprentissage profond dans lesquels on y trouve des odeurs de conceptions. Ses exemples ont été collectés par Amin et al. dans le cadre de leur études empiriques sur les odeurs de conception dans les programmes d'apprentissage profond. En effet, dans le cadre de cette étude, ils ont présenté six exemples de programmes d'apprentissage profond présentant les ordeurs de conceptions énuméré dans l'introduction I. Ces exemples de code source ont été collectés à partir des plateformes StackOverflow et Github.

Le système de détection d'odeurs de conception ainsi développé a servi sur un ensemble de programmes d'apprentissage profond collectés à partir de la plateforme Github. Ces programmes d'apprentissage profond ont été collectés selon un processus autimatiques. En effet, nous avons utilisé l'API de Github et précisement les requêtes de recherche de type *search code*. Cette requête permet de rechercher des fichiers dans les dépôts Github contenant des mots clés spécifiques définis dans notre système. Étant donné que notre papier se concentre uniquement sur les libraires *Keras*, *Tensorflow* et *Pytorch*, nous avons utilisé les mots clés présentés dans le tableau I. Ces mots clés représentent les modules et les fonctions de ces deux librairies.

La requête de recherche de type *search code* retourne un ensemble d'informations sur le repository et le fichier contenant les mots clés recherchés. Nous procédons ensuite à un filtrage des répertoires selon les critères suivants: (1) nombre de commit, (2) notre de star, (3) nombre de fork, (4) dernière date du commit, (5) nombre de contributeurs. Ce filtrage nous permet de ne garder que les répertoires qui sont les plus populaires et qui sont les plus actifs. Nous avons ensuite procédé à un filtrage manuel des répertoires en éliminant les projets qui ne sont pas des programmes d'apprentissage profond.

Table I: *Liste des mots clés utilisés pour la recherche de programmes d'apprentissage profond dans les dépôts Github.*

Mots clés	**Librairie**
keras.layers	Keras
keras.layers.convolutional	Keras
AveragePooling2D	Keras/Tensorflow
MaxPooling2D	Keras/Tensorflow
tensorflow.keras.layers	Tensorflow
Conv2D	Keras/Tensorflow
Convolution2D	Keras/Tensorflow
BatchNormalization	Keras/Tensorflow
import torch	Pytorch
import torchvision.models	Pytorch
torch.nn.Sequential	Pytorch
torch.nn.Conv2d	Pytorch
torch.nn.BatchNorm2d	Pytorch
torch.nn.MaxPool2d	Pytorch

B. Data Processing

In this sub-section, we describe the details of the data collection and analysis approach followed to answer our different research questions. This approach is depicted in Figure **??**. In the following, we elaborate on each data processing step.

1) *Collecter et trier les Smart contract:*
2) *Mesurer le prix du gaz de chaque Smart contrat:* https://github.com/paperSubmition2020/GasmetReplicationPackage
3) *Inserer des modifications (aléatoirement - manuellement):*
4) *Documenter les modifications:*
5) *Mesurer le prix du gaz des smart contrats modifés:*

C. Replication Package

III. Case Study Results

RQ1: Comment détecter les défauts de conception dans les programmes d'apprentissage profond?

RQ2: Quels sont les défauts de conception les plus répandu dans les programmes d'apprentissage profond CNN?

RQ3: Exist-il des lien entre les défauts de conception dans les programmes?

IV. Discussion

V. Threats to validity

This section discusses threats to the validity of our study following the guidelines for case study research.

A. Threats to construct validity

B. Threats to internal validity

C. Threats to External validity

D. Threats to reliability validity

VI. Related Work

A. Replication studies in software engineering

B. Studies on the limitations of SZZ

C. Studies on the impact of Antipatterns

VII. Conclusion

[1]

References

[1] D. Jhon, "test," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 1082–1086.