ESIEE PARIS	Projets Etudiants E4 2023-2024
Version : 1.0	Date : 15/11/2023
Entreprise : DEVERNE SAS 9 Place Marcel Rebuffat 91140 VILLEJUST	
Interlocuteur entreprise : Laurent BOUTIGNY, CTO	
Interlocuteur ESIEE Paris :	

Axe : Entreprise	Titre : Intelligence artificielle pour les caméras embarquées

Domaine(s) du projet :
(les membres de l'équipe doivent appartenir une seule filière)

■ Informatique	☐ Électronique	☐ e-Santé
☐ Data et internet des objets	☐ Systèmes Embarqués	☐ Génie Industriel
☐ Énergie	☐ Réseaux	☐ Sécurité
☐ Communication/Marketing		
☐ Autre :		

MOTS CLES :

ALGORITHMIE, INTELLIGENCE ARTIFICIELLE, EDGE COMPUTING, SYSTEMES EMBARQUES, COMPUTER VISION, CAMERA

CONTEXTE DU PROJET :

Chez Deverne, nous créons les caméras intelligentes et autonomes qui accompagneront l'Homme dans son analyse et sa prise de décision, en toutes circonstances. La caméra intelligente apporte un renouveau dans tous les domaines de la vision : industrie, sécurité, mobilité autonome, défense, dispositifs médicaux…

L'edge IA fait partie des technologies développées chez Deverne depuis 2020. Il s'agit d'une discipline qui s'intéresse à la mise en œuvre d'algorithmes de machine/deep learning sur des calculateurs très contraints en ressources comme les microcontrôleurs.

La pratique usuelle de l'intelligence artificielle nécessite la mise en œuvre de calculateurs à performance moyenne (PC ou serveur physique, cartes graphiques…) ou lourde (réseaux de serveurs en cloud). Cette méthode permet d'exécuter des modèles d'IA de haute précision.

Usuellement, les ingénieurs mettent en place une liaison entre les capteurs (des caméras par exemple) et des serveurs de traitement afin de réaliser un système IA. Ceci pose plusieurs problématiques :

- Sécurité : la donnée transitant entre les capteurs et les serveurs peut être interceptée
- Réactivité : la latence de réponse de l'IA est tributaire de la qualité du réseau de communication entre le capteur et le serveur
- Coûts : la transmission de données brutes vers une centrale IA est coûteuse à cause du volume de données à transmettre

La mise en œuvre de l'IA directement au niveau du capteur permet de répondre à ces enjeux. Dans ce contexte, Deverne emploie des accélérateurs matériels afin de permettre l'exécution en temps réel des algorithmes d'intelligence artificielle.

PROPOSITION DU SUJET :

Le sujet proposé est une étude préliminaire pour la réalisation de l'un de ces accélérateurs matériels. Cette étude **ne nécessite pas de connaissance en électronique ni l'emploi de matériel embarqué** (pas de programmation sur carte de développement).

L'objectif de ce projet est de produire un ensemble de scripts en langage C (permettant le portage en accélérateur matériel) capable d'interpréter un modèle d'intelligence artificielle et d'exécuter ce modèle. Pour cela, nous prendrons l'exemple d'un modèle FOMO, qui permet de reconnaître divers objets dans une image et de les situer.

Pour interpréter et exécuter un modèle d'intelligence artificielle, vous devrez traiter plusieurs étapes :

- Décoder (Parser) le contenu du modèle à partir d'un fichier au format ONNX (format interopérable entre les différents frameworks d'intelligence artificielle)
- Programmer les fonctions unitaires qui composent le modèle d'IA sous la forme d'une bibliothèque C pouvant être enrichie a posteriori.
- Programmer le cœur d'exécution IA en langage C (autrement appelé « interpréteur de modèle ») qui va exécuter les fonctions unitaires selon les instructions du modèle

Voir **Annexe A** à la fin de cette note de cadrage pour plus de détails.

COMPETENCES DEVELOPPEES :

PROGRAMMATION PYTHON : PANDAS, NUMPY...

INTELLIGENCE ARTIFICIELLE : TENSORFLOW, TENSORFLOW LITE, ONNX

ALGORITHMIQUE

PROGRAMMATION C

RETRO ENGINEERING

RESULTATS ATTENDUS :

A L'ISSUE DU PROJET, LES RESULTATS ATTENDUS PAR NOTRE EQUIPE DE DEVELOPPEMENT SONT LES SUIVANTS :

- DISPOSER D'UNE METHODE DE PARSING D'UN MODELE AU FORMAT ONNX ET DE L'INTERPRETEUR DE MODELE PERMETTANT D'EXECUTER LE MODELE
- DISPOSER D'UNE LIBRAIRIE DES FONCTIONS UNITAIRES COMPOSANT LE MODELE FOMO EN C
- DISPOSER D'UNE PROCEDURE DE TEST ET VALIDATION POUR L'INTERPRETEUR DE MODELE ET LES FONCTIONS UNITAIRES
- DISPOSER D'UNE ANALYSE DE LA COMPLEXITE DE L'ALGORITHME ET DE LA MEMOIRE NECESSAIRE A L'EXECUTION DU MODELE EN FONCTION DE DIVERS PARAMETRES DU MODELE (VOIR ANNEXE B A LA FIN DE CETTE NOTE DE CADRAGE POUR PLUS DE DETAILS)

LIVRABLES :

Le projet étant étalé sur 15 jours : J1 à J10 sur 10 semaines, J11 à J15 sur la dernière semaine entière

- J4 : analyse du modèle FOMO, script de lecture de fichier ONNX testé et validé
- J7 : bibliothèque des fonctions unitaires testées et validées
- J10 : script d'interpréteur de modèle non généralisé, testé et validé sur FOMO
- J15 : tous les scripts complets (exécution généralisée d'un modèle IA, librairie de fonctions unitaires et scripts de validation), analyse de complexité et de besoin mémoire, rapport d'étude

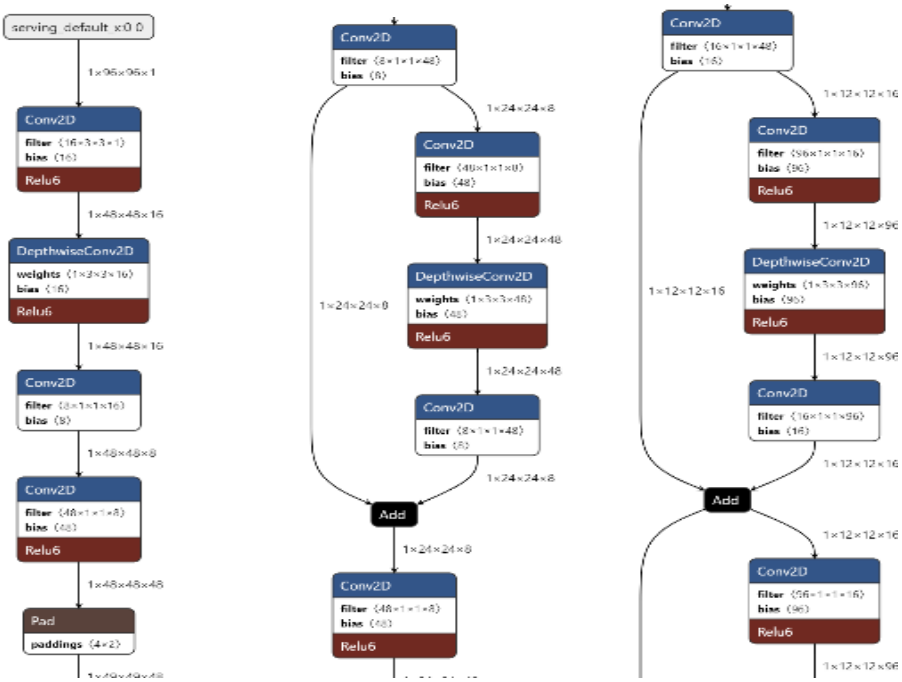
MATERIELS NECESSAIRES :

Deverne mettra à disposition les éléments suivants :

- Modèle FOMO sous la forme d'un fichier au format ONNX
- Base de données de références images pour le test du modèle
- Script Python d'exécution du modèle FOMO sur les références images
- Résultats issus de l'exécution du modèle en Python pour comparaison avec les travaux à effectuer en C

Annexe A : Exemple d'analyse d'architecture de modèle

Ci-dessous, une représentation schématique du modèle FOMO via l'utilisation de l'application d'analyse de modèle Netron (www.netron.app)



Cette représentation sera le point de départ de l'analyse au cours de l'étude proposée. Vous pouvez observer la succession d'opérations mathématiques qui composent le modèle (DeptwiseConv2D, Add, Conv2D)

Note : la figure ci-dessus ne représente **qu'une partie** du modèle

Annexe B : Exemple d'analyse de consommation mémoire

Ci-dessous des exemples de formules que vous aurez à traiter dans le cadre du projet :

Pour déterminer la consommation en mémoire vive lors de l'exécution d'un modèle quantifié 8bits appliqué sur une image 96x96 en nuances de gris, comportant une deux opérations Conv2D chacune comportant un kernel 3x3 et une opération Add on cherche à déterminer quelle est la quantité de stockage en mémoire vive nécessaire pour exécuter toutes les opérations.

On commence par déterminer la taille de l'image d'entrée et l'image de sortie

$$Size\ I_{in} = 96 * 96 * 1 = \ 9\,216\ B$$

$$Size\ I_{out} = 96 * 96 * 1 = \ 9\,216\ B$$

La taille de l'image sortie pour l'opération de Conv2D est identique à celle de l'image d'entrée, on prend aussi en compte la taille du kernel 3x3 utilisé, les poids sont au format 8bits

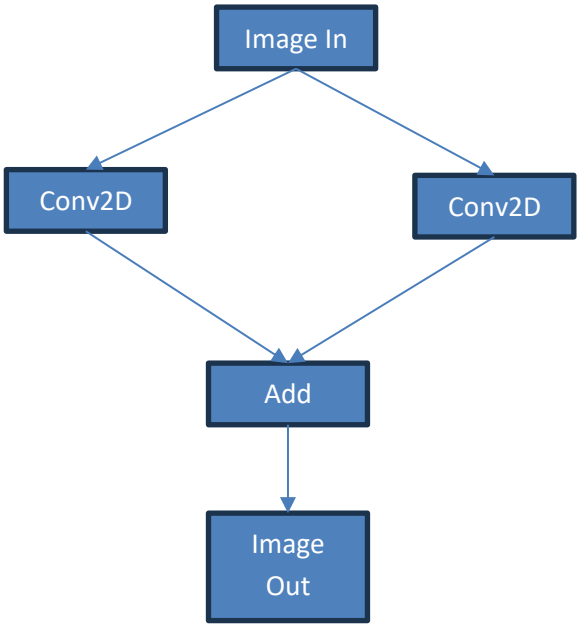
$$Size\ I_{out\ C2D} = 96 * 96 * 1 = \ 9\,216\ B$$

$$Size\ Kernel = 3 * 3 * 1 = \ 9\ B$$

La taille de l'image de sortie de l'opération Add est identique à celle de l'image d'entrée

$$Size\ I_{out\ add} = 96 * 96 * 1 = \ 9\,216\ B$$

On représente le modèle sous par le schéma suivant :



La consommation en mémoire vive est alors :

$$Size\ T = Size\ I_{in} + (Size\ I_{outC2d} + Size\ kernel) * 2 + Size\ I_{outadd} = \sim 37\ kB$$