Application de CoreML à la détection d’intrusion par une caméra

Brice Courault

1. Introduction

L'application Tydom déjà existante détecte les intrusions à l'aide d'une caméra. Celle-ci détecte tout mouvement lié à un franchissement de ligne ou mouvement dans une zone. Le but de ce projet est de faire une étude sur l'intérêt de l'ajout d'un filtre de pertinence de ces alertes. En effet, ces alertes sont envoyées dès lors qu'un mobile déclenche l'un des capteurs de la caméra. Il pourrait donc être intéressant de faire une sélection de ces alertes, afin de notifier à l’utilisateur de l’application seulement les alertes pertinentes, c'est à dire seulement celle qui implique une personne suspecte. Pour cela, une classification à l’aide d’un algorithme de machine learning semble pouvoir être une solution. Un outil mis à disposition par Apple, CoreML, permet d’implémenter un modèle de machine learning sur un device iOS.

Il convient donc de juger de la pertinence de l’implémentation d’une telle solution, de l’intérêt qu’elle suscite pour l’utilisateur, de la faisabilité de cette fonctionnalité et de la justesse des résultats obtenus.

2. Machine learning

Un algorithme à base de machine learning s’implémente sur une application iOS en plusieurs étapes.

- Tout d’abord il y a la mise en place du modèle: énormément de documentations sont disponibles sur des projets open source à ce sujet. À partir des travaux déjà réalisés, sur un environnement python dépendant du modèle choisit, il nous faut compiler ce modèle.
- La seconde étape est la récupération d’une base de données afin de pouvoir

entrainé ce modèle dans le cas où le modèle n'est pas entrainé ou seulement pré-entrainé. Cette base donnée doit être formatée pour correspondre au model utilisé.

- Ensuite, il faut entrainer ce modèle, ce processus nécessite un temps de calcul assez important.
- La quatrième étape est l'exportation de ce modèle afin de l'implémenter sur l'application, sous le format utilisable pas coreML (extension .mlmodel)
- Pour finir, on peut donc utiliser le modèle ainsi généré sur note application en prenant bien soin de formatter les données d'acquisition au format utilisable par le modèle. On peut donc traiter les résultats qui vont être être pondérés par un indice de confiance.

2.1. Choix du modèle

Le choix du modèle est une phase importante car il doit correspondre au cas d'utilisation de celui-ci. Le modèle doit pouvoir traiter les données d'acquisition que l'application va permettre d'utiliser et restituer un résultat qui pourra être utile et utilisable par l'application. Dans notre cas présent, les données d'acquisition sont les captures effectuées par la caméra lors de la détection d'intrusion. Le résultat voulu est un niveau de pertinence de l'alerte intrusion. Des modèles déjà existants permettent de traiter des images telles que celles que nous avons à notre disposition. Nous allons explorer 2 solutions, un modèle permettant la détection d'objets, dans notre cas celle d'une personne, et dans un second temps, la reconnaissance faciale afin de reconnaitre les personnes habitués ne comportant pas de risque pour l'utilisateur.

2.2. Formation de la base de données d'apprentissage

Cette étape est très importante et va conditionner fortement la pertinence des résultats de nos modèles. Dans le cas de la détection de personnes, il convient donc de former une base intéressante d'images permettant au modèle de reconnaitre une

personne dans des contextes variés. Il faut donc constituer un base d'images importante, minimum 300 images par type d'objet que nous voulons détecter afin d'avoir des résultats cohérents.

Dans le cas de la détection de personnes, il convient donc de former une base intéressante d'images permettant au modèle de reconnaitre une personne dans des contextes variés. Il faut donc constituer une base d'images importante, minimum 300 images par type d'objet que nous voulons détecter afin d'avoir des résultats cohérents. Lorsque plusieurs objets peuvent être détectés, il faut que la base d'entrainement soit équitable entre chaque type d'objet, c'est à dire que le nombre d'image représentant chaque type d'objet soit sensiblement les mêmes. De plus pour chaque type d'objet il faut que le contexte, l'environnement et le point de vue sur l'objet soit les plus variés possibles.

Dans le cas de la reconnaissance faciale, **TODO**

3. Différents cas

3.1. Détection de personne

Dans un premier temps, il pourrait être intéressant de pouvoir détecter si l'alerte lancée par la caméra est issu d'un mobile humain. En effet, par exemple si l'alerte a été déclenché par une branche d'un arbre bougeant à cause du vent, d'une voiture qui passe sur la route ou un chat qui traverse le terrain, nous ne voulons pas la notifier à l'utilisateur.

Détecter si une personne est présent au moment de l'alerte dans la zone d'intrusion est un bon moyen de filtrage. Pour cela, une grande bibliothèque de modèles déjà existant développé pour de la détection d'objets, dans notre cas une personne, est disponible. classifier des images, détecter des objets, concevoir des systèmes de recommandations…

4. Développement de la solution

La mise en place du modèle, le formatage de la base d'apprentissage, l'entrainement et l'exportation doivent être effectué côté serveur. L'environnement approprié pour ce type de traitements est le python. Le framework TuriCreate est un framework python. Il convient en plus de mettre en place un webservice associé à ces tâches afin de communiquer avec le reste de l'application. L'entrainement nécessitant un temps important de calcul