UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie robotique

RAPPORT D'APP

Réseaux de neurones convolutifs en traitement d'images
APP2

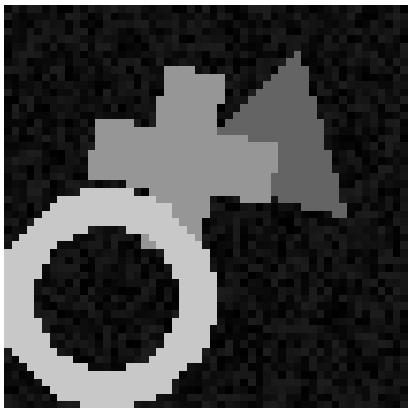
Présenté à
Équipe de tuteurs

Présenté par
Équipe numéro 5
Alexandre Lafleur – lafa3307
Philippe Turcotte – turp2707

Sherbrooke – 18 février 2022

1. INTRODUCTION

La problématique consiste en plusieurs tâches qui doivent être effectuées à partir d'images de formes géométriques dont voici un exemple:

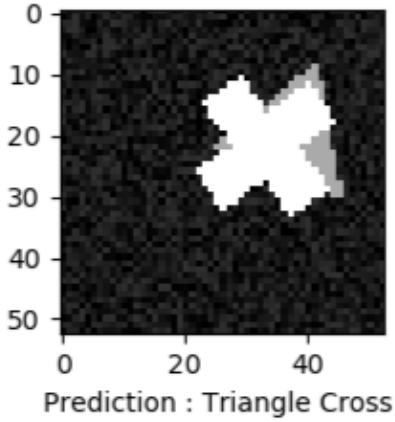


Trois réseau de neurones doivent être entrainés pour effectuer chacun une tâche différente. Voici les tâches qui doivent être accomplies.

2. TÂCHES À ACCOMPLIR

2.1 CLASSIFICATION

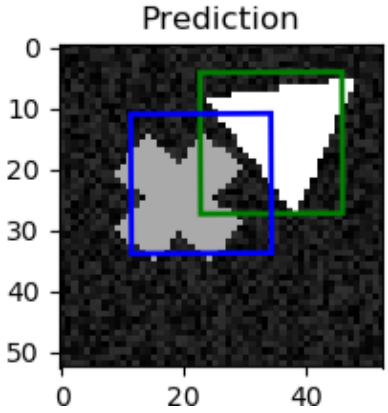
Le réseau doit être capable d'identifier les formes présentes dans une image. Il existe trois formes différentes (triangle, croix, cercle), et il ne peut avoir qu'une seule forme par classse dans l'image. Voici un exemple:



On voit ici une croix ainsi qu'un triangle, et c'est ce que le réseau a prédit. La limite de paramètres pour entrainer ce réseau est de 200 000.

2.2 DÉTECTION

Le réseau doit être capable de créer des boites indiquant la position de chacune des formes dans l'image. Voici un exemple:

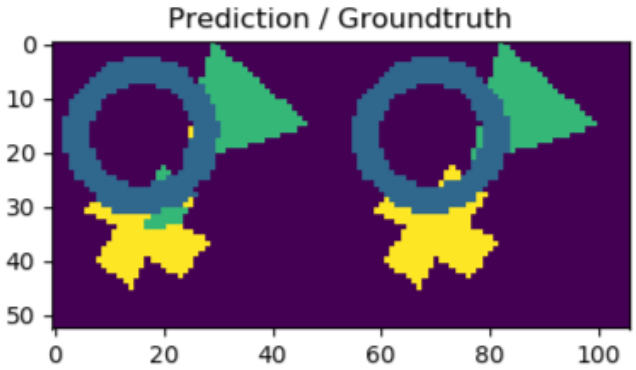


Légende: {Bleu: Croix, Vert: Triangle}

On voit dans l'image que le réseau a plutôt bien prédit la position des formes, en plus de les identifier correctement.

2.3 SEGMENTATION

Le réseau doit être capable d'associer chaque pixel de l'image à sa classe correspondante (arrière-plan, triangle, croix ou cercle). Voici un exemple:



Légende: {Jaune: Croix, Vert: Triangle, Bleu: Cercle, Mauve: Arrière-Plan}

On voit que le réseau (gauche) a correctement segmenté la majorité des pixels, excepté pour une partie de la croix qu'il a pris pour un triangle.

3. DONNÉES

3.1 __LEN__

Le rôle de cette méthod est de retourner le nombre de données contenues dans l'ensemble de donné.

3.2 __GETITEM__

Le rôle de cette méthode est de retourner une image faisant partie des données, ainsi que les objectifs(target) pour l'ensemble des différentes tâches.

3.3 CONVEYORSIMULATOR

Pour utiliser cette classe, il faut lui fournir un *path* vers les images que l'on veut traiter, ainsi que, facultativement, une transformée que l'on veut appliquer à ces images avant le traitement.

3.4 DATALOADER

Le rôle de cette classe est de séparer les données en plusieurs *Datasets* de dimensions égales sur lesquels vont être effectué l'entrainement.

Le paramètre *shuffle* permet de rediviser les données dans des *batchs* différentes à chaque époque, ce qui permet une meilleur distribution des données.

Le paramètre *batch_size* indique le nombre d'images à analyser en même temps par le réseau. Avoir un plus grand nombre d'images permet d'uniformiser et d'accélérer l'apprentissage, mais ce nombre est limité par la mémoire vive que peut utiliser un coeur du processeur.

Le paramètre *num_workers* permet de paralléliser l'apprentissage en créant des sous-processus, ce qui accélère l'apprentissage mais demande une installation particulière sur la machine.

4. RÉSEAUX DE NEURONES

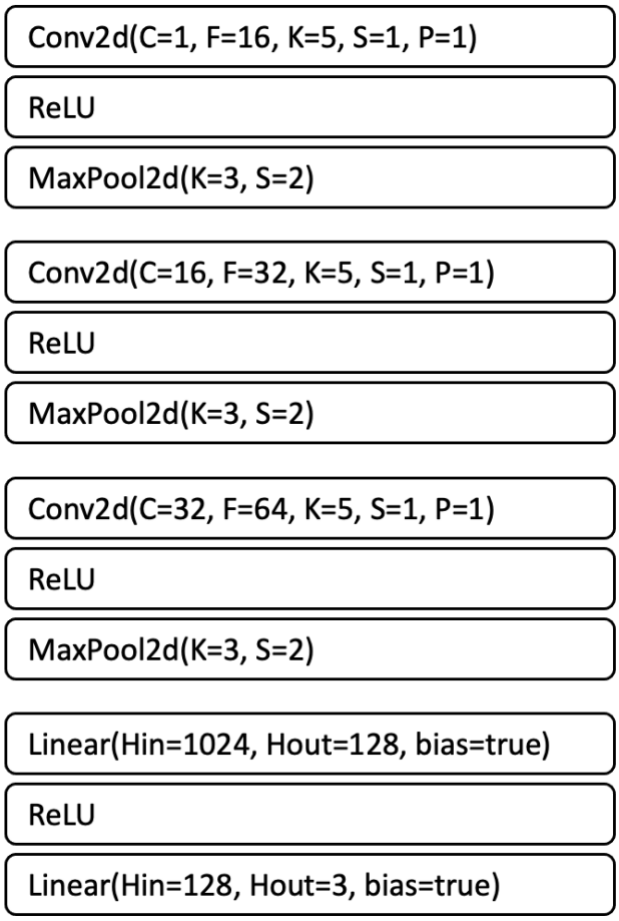
4.1 CLASSIFICATION

4.1.1 ARCHITECTURE

L'architecture choisie se base fortement sur AlexNet, car la combinaison des couches convolutives et de couches linéaires se prête très bien à la classification. Les couches convolutives permettent de faire ressortir les caractéristiques de l'image, et

une fois ces caractéristiques obtenues, les couches linéaires font des liens entre les différentes caractéristiques jusqu'à les associer à des classes. Sans les couches convolutives en premier lieu, il serait beaucoup plus difficile de trouver les caractéristiques d'une image. Cette architecture requiert également très peu de paramètres, comparée à d'autres architecture du même genre, donc c'est un choix idéal pour nous qui devons limiter nos paramètres.

4.1.2 SCHÉMA-BLOC



4.1.3 PARAMÈTRES

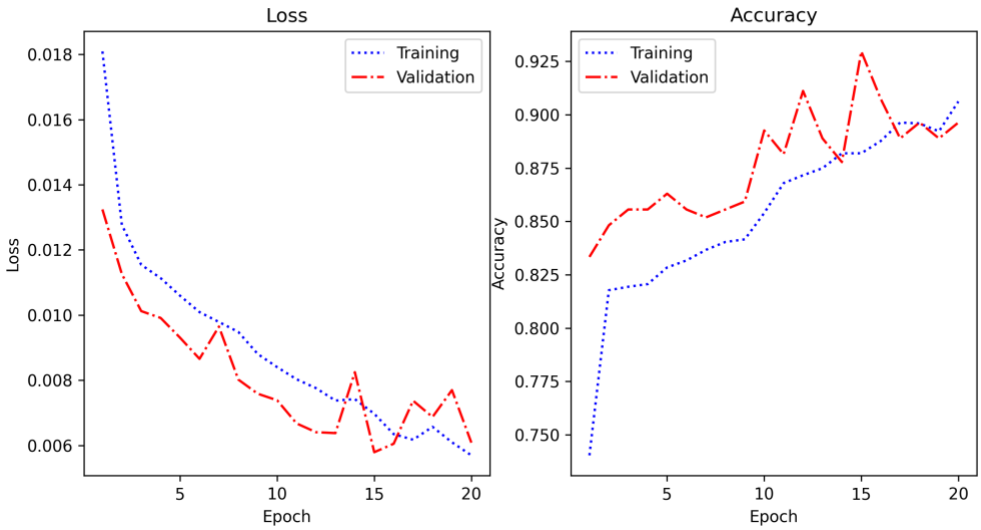
Paramètres classification	Nombre maximum
196 099	200 000

4.1.4 HYPERPARAMÈTRES

La fonction de coût utilisée pour ce réseau est l'entropie croisée avec sigmoide, car il s'agit d'une situation de classification où plusieurs classes peuvent être actives simultanément. Le taux d'apprentissage ainsi que la taille des lots est restée par

défaut, c'est à dire 0.0004 et 32 respectivement. Avec ces hyperparamètres, on l'apprentissage se fait à une vitesse raisonnable et n'est pas trop volatile, du moins pour la partie entrainement.

4.1.5 MÉTRIQUE ENTRAINEMENT ET VALIDATION

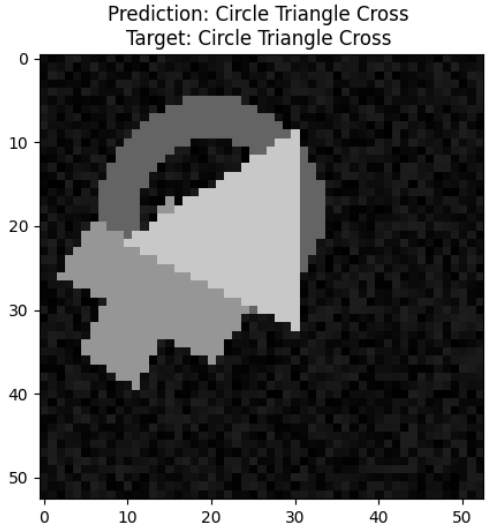


Loss et accuracy pour la tâche de classification

4.1.6 MÉTRIQUE TEST

Loss	Accuracy	Target Accuracy
0.007241	0.923333	0.9

4.1.7 PRÉDICTION

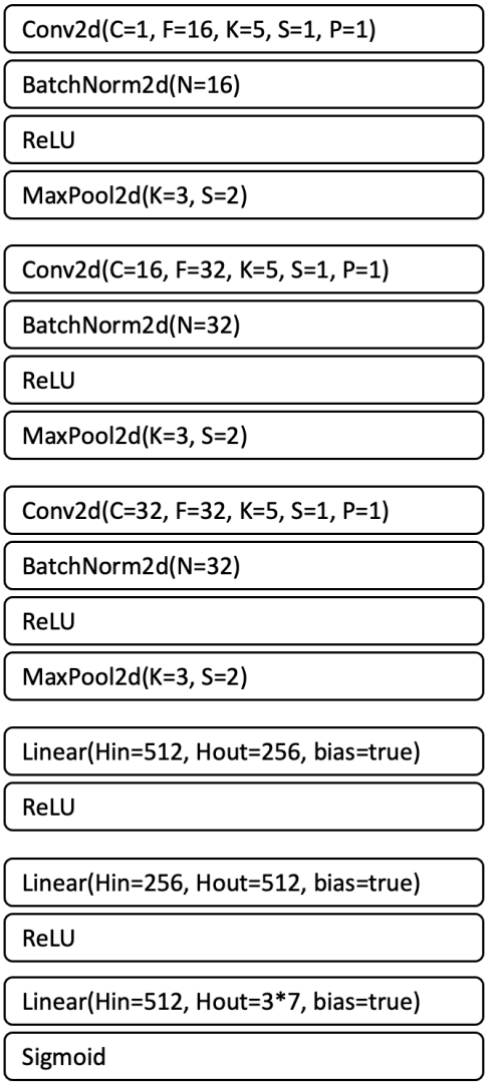


4.2 DÉTECTION

4.2.1 ARCHITECTURE

En premier lieu, l'équipe avait commencé à implémenter YOLO, mais s'est vite rendu compte que cette architecture était beaucoup trop compliquée pour la tâche à effectuer. Puisque les contraintes étaient claires, que seulement 3 types de formes étaient possibles et seulement une de chaque maximum par photo, on pouvait se permettre d'utiliser une architecture moins compliquée, mais autant performante pour notre situation. L'équipe a donc choisi d'utiliser AlexNet, car il est possible de le modifier pour résoudre des problèmes de localisation, et qu'il est très simple, et donc limite le nombre de paramètres utilisés.

4.2.2 SCHÉMA-BLOC



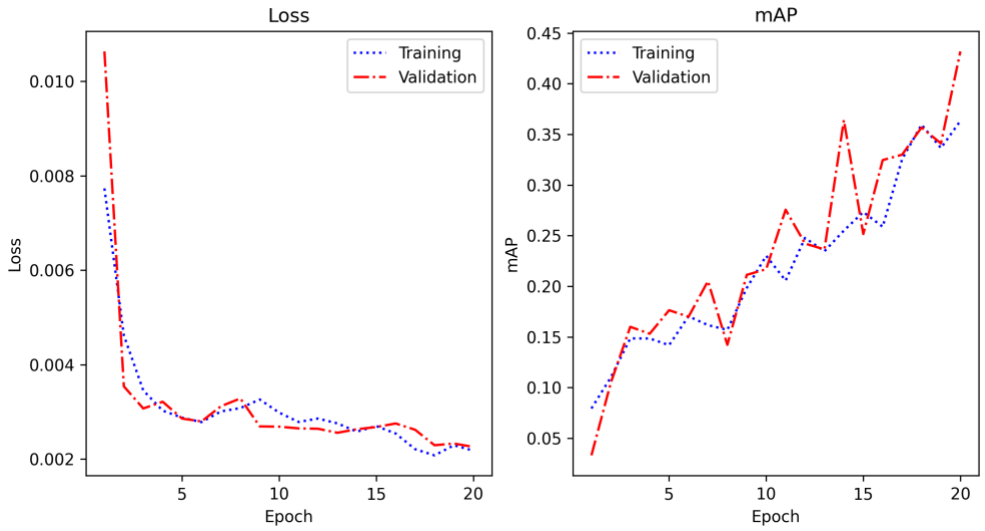
4.2.3 PARAMÈTRES

Paramètres détection	Nombre maximum
312 725	400 000

4.2.4 HYPERPARAMÈTRES

Deux fonctions de coût sont employées dans ce réseau. La première est l'entropie croisée binaire avec sigmoide, qui est utilisée pour la sortie "seuil" ainsi que pour la classification entre les formes. Elle est utilisée car il s'agit de classification et que le seuil empêche d'utiliser un softmax pour les 4 sorties. La deuxième fonction de coût est l'erreur quadratique moyenne, qui est utilisée . Le taux d'apprentissage ainsi que la taille des lots est restée par défaut, c'est à dire 0.0004 et 32 respectivement. Avec ces hyperparamètres, on l'apprentissage se fait à une vitesse raisonnable et n'est pas trop volatile, du moins pour la partie entrainement.

4.2.5 MÉTRIQUE ENTRAINEMENT ET VALIDATION

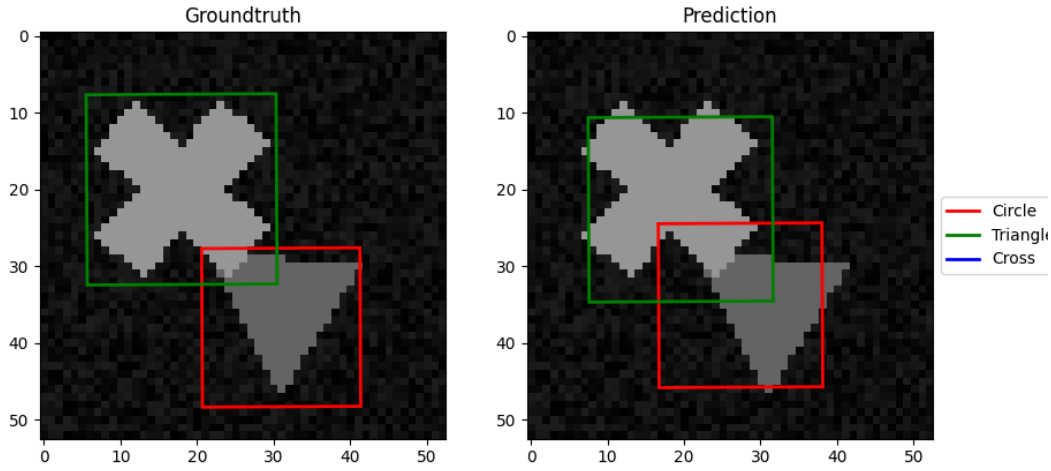


Loss et mAP pour la tâche de détection

4.2.6 MÉTRIQUE TEST

Loss	mAP	Target mAP
0.003392	0.254348	0.3

4.2.7 Prédiction



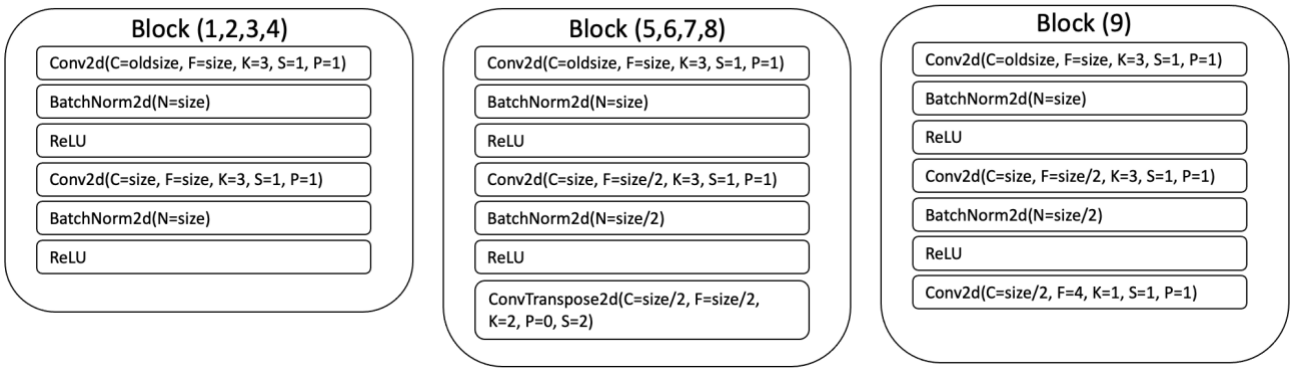
Objectif vs prédiction pour la tâche de détection

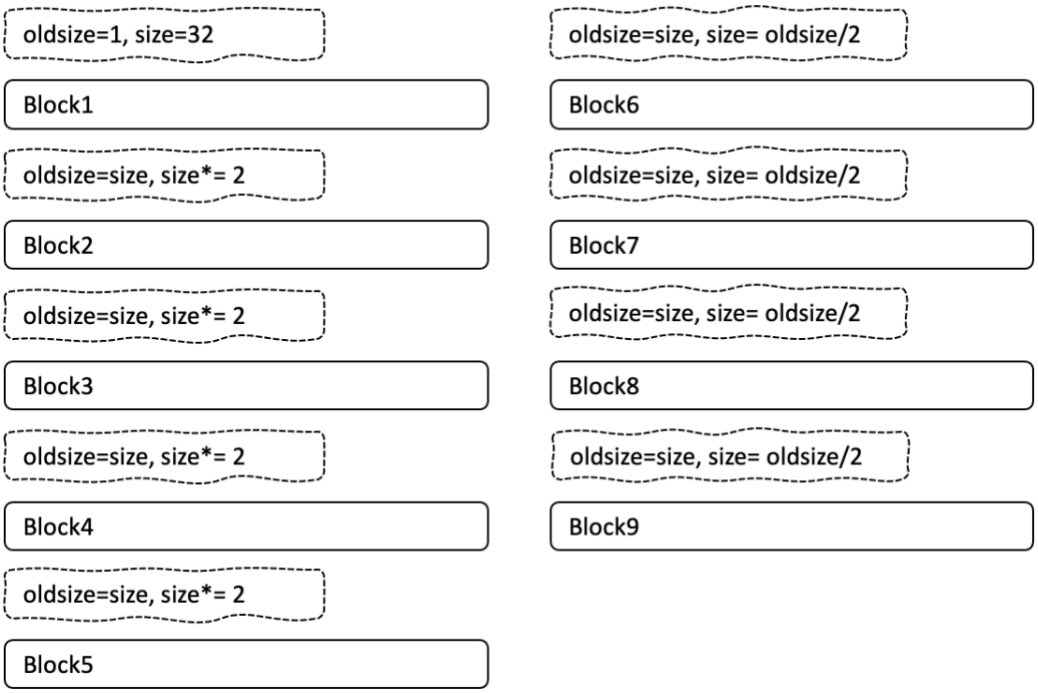
4.3 Segmentation

4.3.1 Architecture

L'équipe a choisi l'architecture U-Net car elle est très bien adaptée à la segmentation. Sa force est qu'elle combine un tenseur avec une bonne résolution spatiale (on s'assure d'avoir le même nombre de pixels en entrée qu'en sortie), et un tenseur de basse résolution, mais qui est utilisé pour trouver les caractéristiques pertinentes aux différentes classes. En les combinant, on obtient la parfaite architecture de réseau, car on peut trouver chaque pixel appartient à quelle classe.

4.3.2 Schéma-bloc





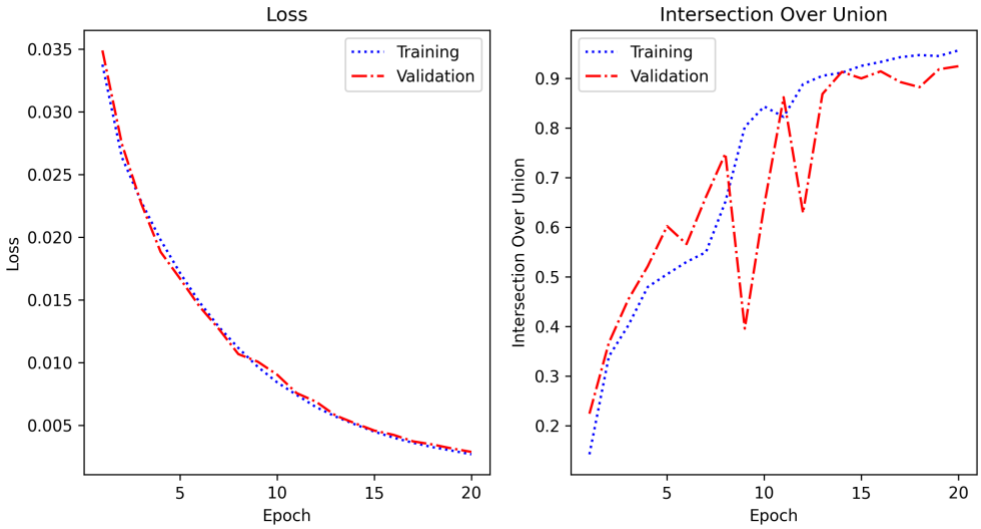
4.3.3 PARAMÈTRES

Paramètres segmentation	Nombre maximum
823 798	1 000 000

4.3.4 HYPERPARAMÈTRES

La fonction de coût utilisée pour ce réseau est l'entropie croisée avec softmax, car il s'agit d'une situation de classification où une seule classe peut être active à la fois. Le taux d'apprentissage ainsi que la taille des lots est restée par défaut, c'est à dire 0.0004 et 32 respectivement. Avec ces hyperparamètres, on l'apprentissage se fait à une vitesse raisonnable et n'est pas trop volatile, du moins pour la partie entrainement.

4.3.5 MÉTRIQUE ENTRAINEMENT ET VALIDATION

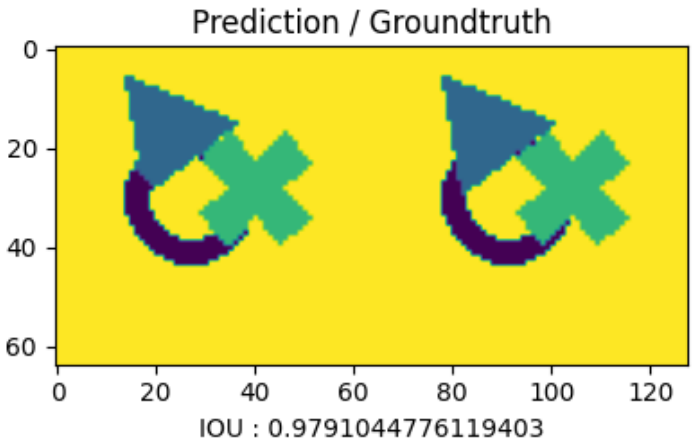


Loss et IOU pour la tâche de segmentation

4.3.6 MÉTRIQUE TEST

Loss	IOU	Target IOU
0.003678	0.91632	0.9

4.3.7 PRÉDICTION



Prédiction vs Objectif pour la tâche de segmentation

5. CONCLUSION

5.1 CLASSIFICATION

Les performances du réseau de classification sont assez satisfaisantes. En effet, la précision lors du test est de 92% ce qui est supérieur à la valeur cible de 90% suggérée pour ce réseau. De plus, la précision lors de la validation stagnait lors de l'entrainement, ce qui signifie que le réseau était suffisament entrainé pour sa tâche. Si on souhaitait avoir d'encore meilleur résultats, des normalisations de lots pourraient être rajouté aprés les couches de convolutions et les couches linéaires.

5.2 DÉTECTION

Les performances du réseau de détection laissent à désirer. En effet, la *mean average precision* lors du test n'est que de 0.25, en dessous de la valeur cible de 0.3. Cela peut être dû au fait que la *binary cross entropy* avec sigmoide est utilisée comme fonction de coût pour la classification, alors que qu'une entropie croisée standard avec *softmax* aurait été plus appropriée.

5.3 SEGMENTATION

Les performances du réseau de segmentation sont très satisfaisantes. En effet, l'intersection sur l'union lors de test était de 0.91, soit au dessus de la valeur cible de 0.9. De plus, dans l'exemple illustré de la prédiction, le IOU est de 0.97 malgré le fait que l'image n'est pas simple à segmenter. Nous avions des résultats encore meilleur avec davantages de paramètres, mais nous avons dû les limiter pour respecter les contraintes de la problématique.