Projet de fin de session

Image captionning

Travail remis à M. Pascal Germain
dans le cadre du cours
GLO-4030 - *Apprentissage par réseaux de neurones profonds*

par

Ilies Benhaddouche (111 236 419)
ilies.benhaddouche.1@ulaval.ca

Département d'informatique et de génie logiciel
Faculté des sciences et de génie, Université Laval
Mai 2021

1 Résumé de la tâche et motivation

Le but de ce travail est de construire un système qui permet de générer un texte descriptif à partir d'une image passée en entrée (i.e un système d'image captionning).

Ce sujet a été choisi, parce qu'à travers sa réalisation, il sera possible de pratiquer en profondeur certaines notions de deep learning évoquées de façon théorique dans le cours.

Ainsi, **l'implémentation du système à partir de zéro**, permettra de :

— Mieux comprendre la notion de Sequence-To-Sequence.

— Avoir un aperçu sur l'utilisation du Word Embedding de façon précise et du traitement de la langue de façon générale.

— Mieux comprendre le fonctionnement d'un Encodeur-Décodeur

— Mieux comprendre le fonctionnement des LSTM.

D'autre part, le projet permettra de revenir sur d'autres notions déjà traitées lors des TP, mais dans un contexte différent, tel que le fine tuning avec les réseaux de neurones convolutifs.

2 Introduction

Pour arriver à générer du texte descriptif à partir d'une image, il va falloir en premier lieu encoder notre image pour récupérer ce qu'elle contient comme caractéristiques, puis dans un second lieu, traduire ces dernières en une sorte de plongements de mots qui seront décodés, ce qui permettra au finale d'avoir du texte descriptif.

3 Stratégie pour résolution de la tâche

La résolution se fera en deux étapes :

1. La première étape consistera à extraire les features d'une image. Cet encodage se fera à l'aide d'un réseau de neurone convolutif pré-entraîné, auquel la couche de prédiction (softmax) aura été préalablement retirée.

2. Une fois les caractéristiques extraites, il sera question de les décoder à l'aide d'un modèle Sequence, tel que GRU ou LSTM, mais pour cette exercice c'est la LSTM qui a été retenue.

Donc, il faudra concevoir une partie "Encodeur" et une autre partie "Décodeur" qui pourront être connectées de (2) deux manières possibles :

— La première consiste à fournir le vecteur features comme premier input au réseau décodeur. Si c'est le cas, on voudrait avoir le token "<Beginning Of Sentence>", comme premier output du premier timestep.

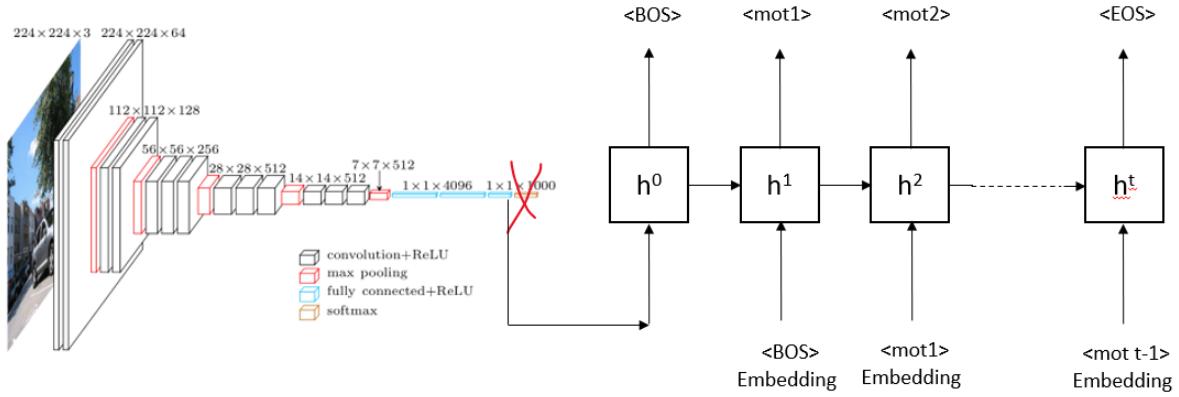


FIGURE 1 – Première façon pour connecter encodeur et décodeur

— La deuxième consiste à fournir le vecteur features comme premier état caché au réseau décodeur. Si c'est le cas on devrait fournir le token "<Beginning Of Sentence>", comme premier input.

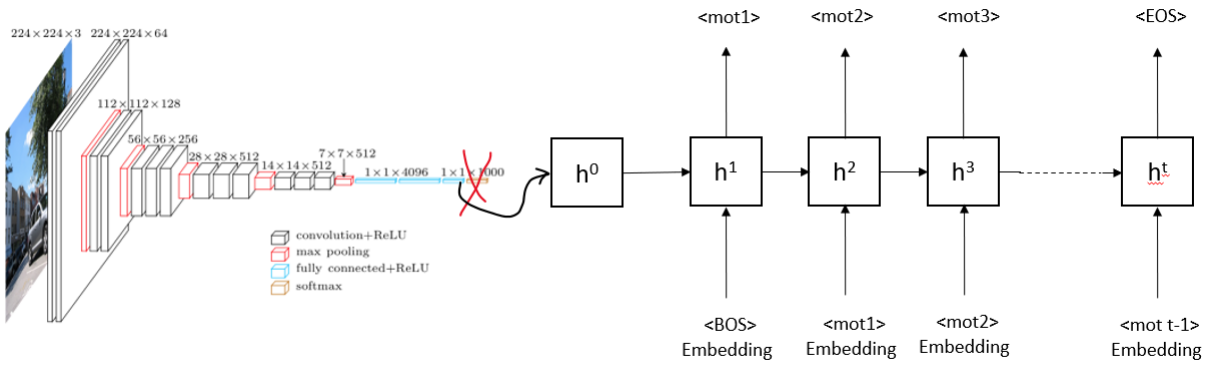


FIGURE 2 – Deuxième façon pour connecter encodeur et décodeur

Pour cet exercice, on utilisera la première façon de se connecter, i.e en fournissant le vecteur features au réseau décodeur, comme premier input.

4 Architecture

4.1 Encodeur

Le réseau convolutif pré-entraîné choisi, afin d'extraire les features de nos images, est le réseau "**Inception Net**". Ce choix est motivé par le fait que ce réseau enregistre généralement de très bons scores de performance et cela est dû essentiellement à son bloc de convolution qui se retrouve au coeur de son architecture, ainsi qu'a son nombre élevé de paramètres.

Puisque pour cet exercice, il sera question de faire du fine tuning avec ce CNN, on procédera à la suppression de la couche de prédiction(softmax) et en plus, juste après avoir récupérer les features de l'image, récoltées sur la dernière couche fully connected, on rajoute une autre couche linéaire dans le seul but est d'adapter les dimensions du vecteur features à celles de l'entrée du décodeur.

Ainsi la sortie de la dernière couche linéaire, rajoutée, devrait correspondre à la taille de l'embedding (embedding size), utilisé par le décodeur.

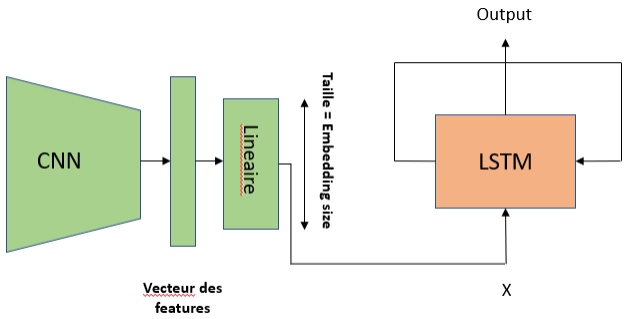


FIGURE 3 – Re-dimension de la sortie de l'encodeur

En outre, tout de suite après la dernière couche linéaire, on ajoute une couche de dropout avec une probabilité de 0.5, ce qui permettra de faire de la régularisation.

Ainsi, les toutes dernières couches de l'encodeur se superposeront comme suit :

```
  (fc): Linear(in_features=2048, out_features=256, bias=True)
)
(relu): ReLU()
(dropout): Dropout(p=0.5, inplace=False)
```

FIGURE 4 – Dernières couches de l'encodeur

4.2 Décodeur

Le décodeur est constitué de la superposition des couches suivantes :

— Une couche d'embedding, car on a besoin d'avoir les plongement des mots avant de les fournir comme inputs. (Cette couche est de dimension : Taille du vocabulaire x Taille de l'embedding).

— Une couche dropout, d'un taux d'inclusion de 0.5, qui sera appliquée à la sortie de la couche embedding.

— Une couche LSTM One to many. Par souci d'entraînement, la nombre de couches choisi pour le réseau LSTM est égal à 1 et le nombre d'états cachés égal à la taille de l'embedding.

— une couche linéaire pour ajuster la dimension de sortie qui doit être égale à la taille du vocabulaire (Dimension de la couche : Taille des états cachés x Taille du vocabulaire)

```
(decoderRNN): DecoderRNN(
  (embed): Embedding(1000, 256)
  (lstm): LSTM(256, 256)
  (linear): Linear(in_features=256, out_features=1000, bias=True)
  (dropout): Dropout(p=0.5, inplace=False)
)
```

FIGURE 5 – Couches du décodeur

5 Optimisation / Régularisation du réseau

— **Fonction de perte** : La fonction de perte utilisée et la Cross Entropy. Il est à noter que pour le token <PAD>, le calcul de perte ne doit pas se faire.

— **Fonction d'optimisation** : Adam.

— **Fonction d'activation** pour la dernière couche linéaire de l'encodeur est la relu suivie d'une **régularisation** dropout (probabilité de 0.5).

— Une **régularisation** dropout (probabilité de 0.5) à la sortie de la couche d'embedding du décodeur

6 Méthodologie et procédure d'entraînement :

6.1 Le jeu de données :

Pour les besoins du projet, il a été utilisé le jeu de données **Flickr 8k Dataset**, téléchargé sur kaggle : https ://www.kaggle.com/adityajn105/flickr8k

Le jeu contient 8091 images et pour chaque image, (5)cinq textes descriptifs (captions) lui sont associés. Les cinq expriment, bien sûr, la même idée. Exemple :


```
166321294_4a5e68535f.jpg,A man is riding on a red motorcycle .
166321294_4a5e68535f.jpg,A motorcycle driver dressed in orange gear swerves to the right .
166321294_4a5e68535f.jpg,A motorcyclist on a red speed bike leans into a sharp turn .
166321294_4a5e68535f.jpg,Motorcyclist crouches low as he rounds a turn .
166321294_4a5e68535f.jpg,This person is on a red motorcycle .
```

FIGURE 6 – Paire : image - captions

Le dataset a été splitté en deux parties. Une première de 60% pour le jeu d'entraînement et une deuxième de 40% pour le jeu de test.

6.2 Étapes d'entraînement

6.2.1 Construction du vocabulaire

La première étape consistait à construire le vocabulaire à partir des captions du jeu de données dédié pour l'entraînement.

Pour ce faire, il a été fixé pour les mots un seuil minimal de fréquence (Frequency Threshold), en dessous duquel un mot donné ne fera pas partie du vocabulaire, car considéré peu important. Pour l'exercice, il a été considéré que tous les mots sont importants et par conséquent la valeur 1 a été attribuée au seuil minimal de fréquence.

Pour avoir un aperçu, un vocabulaire pourrait ressembler à ce qui suit :

vocabulaire = {0 :<PAD>, 1 :<SOS>, 2 :<EOS>, 3 :<UNK>, 4 :a, 5 :dog', 6 :the,...}

Le vocabulaire sera utilisé pour numériser le texte, censé être fourni au décodeur. Ainsi, les items d'une batch seront des paires : images - captions numérisées.

En outre, un padding a été appliqué pour ces captions cibles, afin qu'il aient la même longueur de séquence.

6.2.2 Extraction de features

Avant d'envoyer une image au CNN, un traitement préalable a été appliqué :

— Redimension 356 x 356 et recadrement aléatoire.

— Normalisation en utilisant les valeurs utilisées pour entraîner InceptionNet : RGB=(0.5, 0.5, 0.5), Écart-type=(0.5, 0.5, 0.5).

Tous les paramètres du CNN ont été gelés, sauf ceux de la fully connected, où les caractéristiques de l'image seront récoltées.

6.2.3 Génération du texte descriptif

L'entrée du décodeur reçoit , comme mentionné ci-haut, des paires : vecteur features - caption numérisé.

Le vecteur features, sera passé comme premier input (premier timestep) à la couche LSTM. Pour les timesteps suivants, les Words Embedding, constituant la légende seront passés au LSTM, de façon séquentielle.

Comme illustré, à la figure 1, à la fin on récupère un texte descriptif.

6.3 Note concernant l'inférence (Test)

Il est à noter, ici, l'existence d'une petite différence dans la façon de procéder, quand il s'agira de faire de l'inférence.

En effet, quand on fait de l'inférence, la sortie d'un état précédent (t-1) sera toujours utilisée comme input pour l'état suivant (t).

Par contre pour l'entraînement, ce n'était pas le cas, car si jamais la sortie d'un état précédent est erronée, on ne voudrait pas utilisé ce résultat biaisé comme input pour l'état suivant. Pour cette raison, lors de la phase d'entraînement, le réseau LSTM sera toujours alimenté avec les vrais mots cibles et non ceux prédits par LSTM à chaque timestep.

6.4 Note concernant le calcul de la précision

Vu la nature du problème à résoudre, le seul moyen de comparer les prédictions avec les cibles, serait de les soumettre au jugement humain, car pour rappel, il sera question, ici, de comparer le sens global de deux textes descriptifs.

Après plusieurs recherches, je n'ai pas trouvé une métrique qui pourrait remplacer le jugement d'un humain.

Toutefois, des diagrammes de perte seront générés, afin de suivre la performance du modèle.

7 Résultats des expérimentations et interprétation :

Contexte d'entraînement :

Étant donné, les temps trop longs, requis pour les entraînements, ainsi que les quotas d'utilisation imposés par Kaggle et Google Colab, je n'ai pu faire autant d'entraînements que je le voulais (tester plusieurs combinaisons d'hyper-paramètres, test de plusieurs optimisateur...etc).

Quant au site de Calcul Québec, je n'ai pu l'utiliser, car je n'ai pas réussi à installer toutes les librairies (telle que Spacy) requises pour le projet sur l'environnement créé à cet effet.

Ce qui fait, l'entraînement a eu lieu dans le seul contexte suivant :

Number of epochs	Batch size	Learning rate	Embedding Size
80	32	0.0003	256

Hidden size LSTM	Number Layer LSTM	Frequency Threshold	Probability - Dropout
256	1	1	0.5

7.1 Résultats pour le jeu d'entraînement :

7.1.1 Pertes du modèle

Après 80 epochs, la perte enregistrée avait une valeur de **1.728**. Ce qui fait, avec plus d'entraînement, il aurait été possible de minimiser la perte davantage.

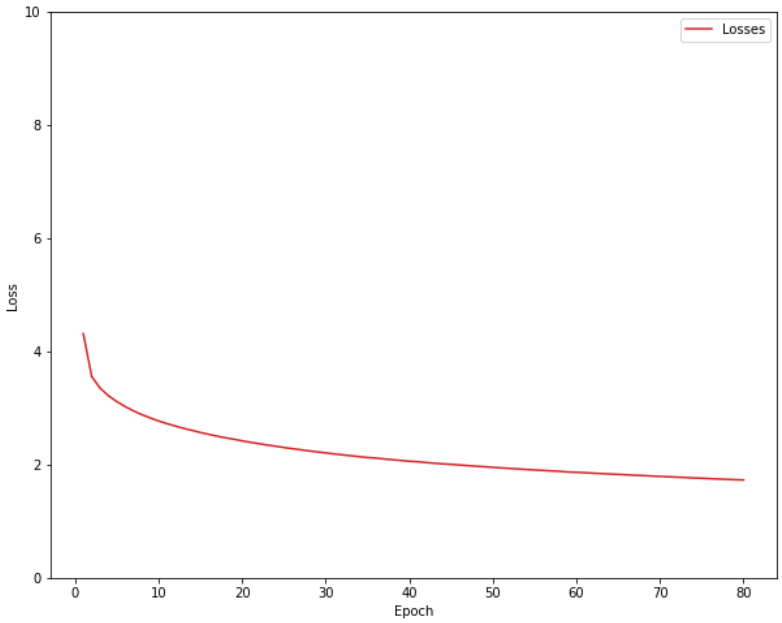


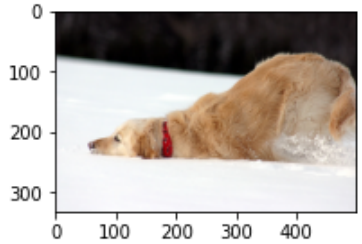
FIGURE 7 – Pertes du modèle

7.1.2 Performance du modèle

Dix (10) observations ont été pris au hasard afin de vérifier la performance sur le jeu d'entraînement. Ci-dessous, les résultats pour trois, le reste est en annexe :

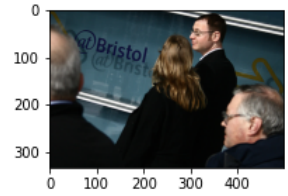
```
Annotation cible: Three people drive ATVs .

Annotation prediction: ['<SOS>', 'three', 'people', 'are', 'riding', 'a', 'blue', 'atv', '.', '<EOS>']
```

Une erreur concernant la couleur du vtt, mais sinon bonne prédiction.



```
Annotation cible: A brown dog has his face down in the snow .

Annotation prediction: ['<SOS>', 'a', 'dog', 'runs', 'through', 'the', 'snow', '.', '<EOS>']
```

La prédiction n'est pas aussi précise que la cible, mais elle est tout de même satisfaisante.

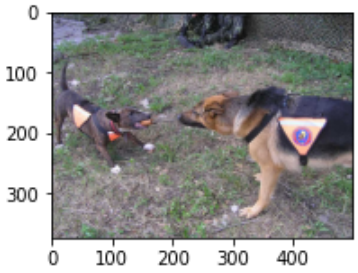


```
Annotation cible: A group of adults are walking .

Annotation prediction: ['<SOS>', 'a', 'man', 'and', 'a', 'woman', 'are', 'sitting', 'on', 'a', 'couch', '.', '<EOS>']
```

La prédiction est mauvaise. Le modèle a réussi à distinguer la présence d'une femme et d'un homme, par contre le reste du contexte n'est pas bon.

7.2 Résultats pour le jeu de test :

Dix (10) observations ont été pris au hasard afin de vérifier la performance sur le jeu de test. Ci-dessous, les résultats pour trois, le reste est en annexe :

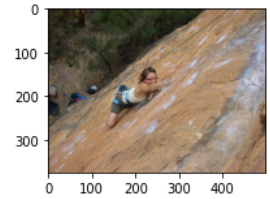


```
Annotation cible: A small black dog is playing tug of war with a large brown dog wearing a yellow triangle .

Annotation prediction: ['<SOS>', 'two', 'dogs', 'are', 'playing', 'in', 'a', 'field', '.', '<EOS>']
```

La prédiction est très satisfaisante.

```
Annotation cible: An empty seat at a Mexican themed restaurant .

Annotation prediction: ['<SOS>', 'a', 'group', 'of', 'people', 'are', 'standing', 'around', 'a', 'sound', 'mixing', 'table', '.', '<EOS>']
```

La prédiction n'est pas bonne. Le modèle a réussi à distinguer la présence d'un groupe de personnes, mais le reste du contexte est mauvais.



```
Annotation cible: A girl wearing glasses is in a blue harness while rock climbing .

Annotation prediction: ['<SOS>', 'a', 'man', 'in', 'a', 'red', 'jacket', 'is', 'standing', 'on', 'a', 'rock', 'overlooking', 'a', 'valley', '.', '<EOS>']
```

Très mauvaise prédiction. Contexte totalement faux.

8 Conclusion

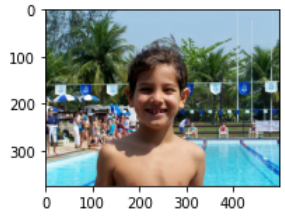
De façon générale, le modèle a réussi à apprendre correctement et vu les résultats obtenus sur les jeux de test et d'entraînement, on peut conclure que le modèle n'as pas fait du sous-apprentissage et qu'il généralise pas mal. Par contre, il est claire qu'il requérait plus d'entraînement et un vocabulaire plus fourni.

En effet, théoriquement avec plus d'entraînement et une meilleur sélection des hyper-paramètres, il aurait été certainement possible d'avoir une meilleur optimisation de perte et par conséquent de meilleurs résultats. Mais comme mentionné, plus haut, pour des raisons techniques, cela n'a pas été possible.

D'une autre part, il aurait été plus intéressant d'ajouter un modèle d'attention visuelle pour améliorer la performance.

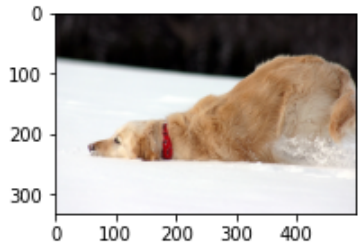
9 Annexe

9.1 Résultats pour le jeu d'entraînement



Annotation cible: A boy is smiling whilst standing in front of a swimming pool .

Annotation prediction: ['<SOS>', 'a', 'young', 'boy', 'wearing', 'a', 'blue', 'shirt', 'is', 'walking', 'on', 'a', 'beach', '.', '<EOS>']

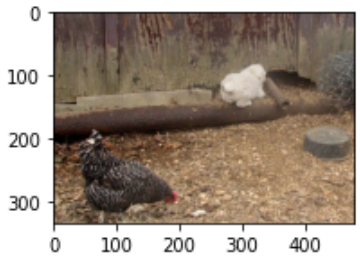


Annotation cible: A brown dog has his face down in the snow .

Annotation prediction: ['<SOS>', 'a', 'dog', 'runs', 'through', 'the', 'snow', '.', '<EOS>']

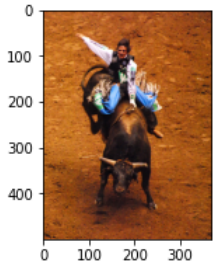
Annotation cible: Three people drive ATVs .

Annotation prediction: ['<SOS>', 'three', 'people', 'are', 'riding', 'a', 'blue', 'atv', '.', '<EOS>']

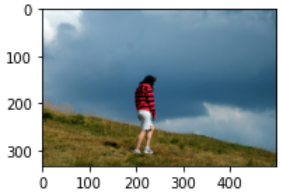


Annotation cible: A chicken and a white dog in the mulch .

Annotation prediction: ['<SOS>', 'a', 'dog', 'is', 'running', 'through', 'the', 'water', '.', '<EOS>']



Annotation cible: "A bull rider hangs on tightly to a rope around a large
Annotation prediction: ['<SOS>', 'a', 'man', 'in', 'a', 'red', 'shirt', 'is', 'riding', 'a', 'bike', 'on', 'a', 'dirt', 'path', '.', '<EOS>']

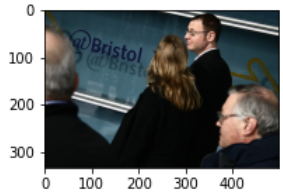


Annotation cible: A person in a red striped hooded sweatshirt and jeans shorts walking on a grassy hill .

Annotation prediction: ['<SOS>', 'a', 'man', 'in', 'a', 'red', 'shirt', 'is', 'climbing', 'a', 'rock', 'wall', '.', '<EOS>']

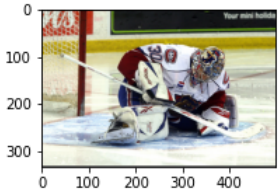
Annotation cible: A man is in the air with his bike and there 's another guy on his bike in the back with the water close by .

Annotation prediction: ['<SOS>', 'a', 'man', 'on', 'a', 'motorcycle', 'is', 'turning', 'a', 'dirt', 'bike', '.', '<EOS>']



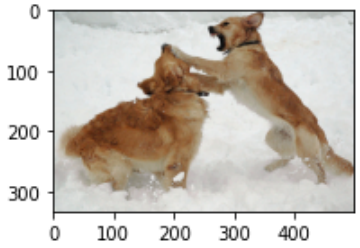
Annotation cible: A group of adults are walking .

Annotation prediction: ['<SOS>', 'a', 'man', 'and', 'a', 'woman', 'are', 'sitting', 'on', 'a', 'couch', '.', '<EOS>']



Annotation cible: A goalie blocks the puck while holding a white hockey stick .

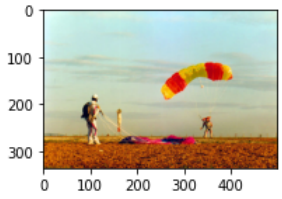
Annotation prediction: ['<SOS>', 'a', 'hockey', 'player', 'in', 'red', 'untangles', 'from', 'a', 'goal', '.', '<EOS>']



Annotation cible: A brown and white dog bares his teeth and grabs the head of another brown and white dog .

Annotation prediction: ['<SOS>', 'two', 'dogs', 'are', 'playing', 'in', 'the', 'snow', '.', '<EOS>']

9.2 Résultats pour le jeu de test

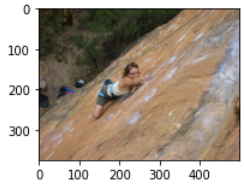


Annotation cible: A skydiver safely lands while another watches from the ground .

Annotation prediction: ['<SOS>', 'a', 'man', 'is', 'standing', 'on', 'a', 'rock', 'overlooking', 'a', 'lake', '.', '<EOS>']

Annotation cible: A lady stands in the middle of a crowd wearing white gloves .

Annotation prediction: ['<SOS>', 'a', 'woman', 'in', 'a', 'blue', 'shirt', 'is', 'sitting', 'on', 'a', 'wooden', 'bench', '.', '<EOS>']

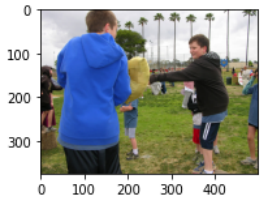


Annotation cible: A girl wearing glasses is in a blue harness while rock climbing .

Annotation prediction: ['<SOS>', 'a', 'man', 'in', 'a', 'red', 'jacket', 'is', 'standing', 'on', 'a', 'rock', 'overlooking', 'a', 'valley', '.', '<EOS>']

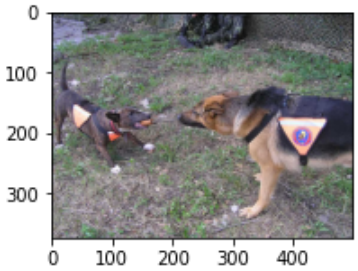
Annotation cible: An empty seat at a Mexican themed restaurant .

Annotation prediction: ['<SOS>', 'a', 'group', 'of', 'people', 'are', 'standing', 'around', 'a', 'sound', 'mixing', 'table', '.', '<EOS>']



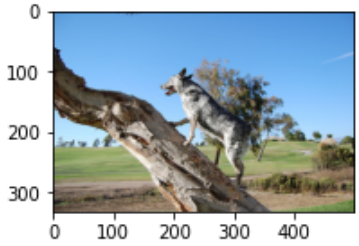
Annotation cible: A boy in a black shirt punches a yellow punching bag while a boy in a blue sweatshirt holds the bag .

Annotation prediction: ['<SOS>', 'a', 'woman', 'in', 'a', 'white', 'dress', 'walks', 'with', 'a', 'man', 'in', 'a', 'black', 'jacket', '.', '<EOS>']



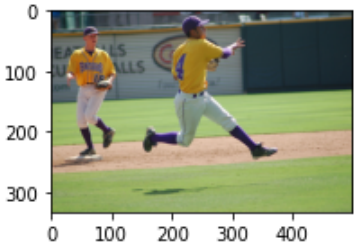
Annotation cible: A small black dog is playing tug of war with a large brown dog wearing a yellow triangle .

Annotation prediction: ['<SOS>', 'two', 'dogs', 'are', 'playing', 'in', 'a', 'field', '.', '<EOS>']



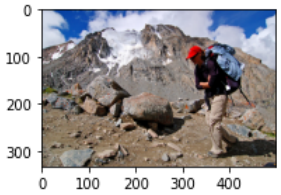
Annotation cible: A black and white dog climbs a leaning tree trunk .

Annotation prediction: ['<SOS>', 'a', 'man', 'in', 'a', 'red', 'shirt', 'is', 'snowboarding', '.', '<EOS>']



Annotation cible: Men play baseball .

Annotation prediction: ['<SOS>', 'two', 'men', 'playing', 'soccer', '.', '<EOS>']



Annotation cible: A backpacker is walking in front of a mountain with arms crossed .

Annotation prediction: ['<SOS>', 'a', 'man', 'is', 'riding', 'a', 'mountain', 'bike', 'on', 'a', 'dirt', 'path', '.', '<EOS>']

Annotation cible: A kid skateboards down railings of a large set of stairs .

Annotation prediction: ['<SOS>', 'a', 'man', 'in', 'a', 'red', 'shirt', 'is', 'doing', 'a', 'trick', 'on', 'a', 'skateboard', '.', '<EOS>']