Image Captioning avec RNN et Attention sur ResNet

Mohammed BARDAOUI

Master SDIA - Data Science et Intelligence Artificielle

Introduction

Le TP *Attention : Image Captioning avec RNN et Attention sur ResNet* vise à implémenter un modèle capable de générer des légendes pour des images en combinant un ResNet pré-entraîné, utilisé pour extraire les caractéristiques visuelles, avec un RNN doté d'un module d'attention afin de produire des captions textuelles pertinentes. L'objectif est de transformer une image en une description textuelle appropriée en utilisant le dataset *Flickr30k*, qui contient plus de 30 000 images annotées avec des descriptions. Ce TP permet de se familiariser avec la manipulation d'un dataset d'image captioning, le chargement et la transformation des données, le transfert d'apprentissage via un ResNet50 pré-entraîné et le gel de ses paramètres. Il inclut également l'implémentation d'un module d'attention personnalisé, la création d'une architecture LSTM modifiée avec attention, l'utilisation d'embeddings pré-entraînés (*Word2Vec*) pour représenter les séquences textuelles, ainsi que la maîtrise de la tokenisation et détokenisation pour la manipulation des séquences. Enfin, le TP permet de concevoir une boucle d'entraînement incluant une planification du taux d'apprentissage (*Step Decay*) et d'évaluer l'évolution du modèle en générant des légendes pour les images au fil des époques.

1 Chargement et prétraitement des données

Le dataset *Flickr30k* contient plus de 30 000 images annotées avec plusieurs descriptions textuelles. Les étapes de préparation sont les suivantes :

- Nettoyage des noms de colonnes et des textes dans le fichier CSV des annotations.
- Redimensionnement des images à 224×224 pixels pour être compatibles avec ResNet50.
- Conversion des images en tenseurs PyTorch.
- Séparation du dataset en ensembles d'entraînement (80%) et de test (20%).
- Création de *DataLoaders* avec une taille de batch de 32 pour faciliter l'entraînement.

Visualisation du dataset

Pour mieux comprendre le dataset, nous présentons ci-dessous un aperçu général et un exemple d'image avec sa légende.

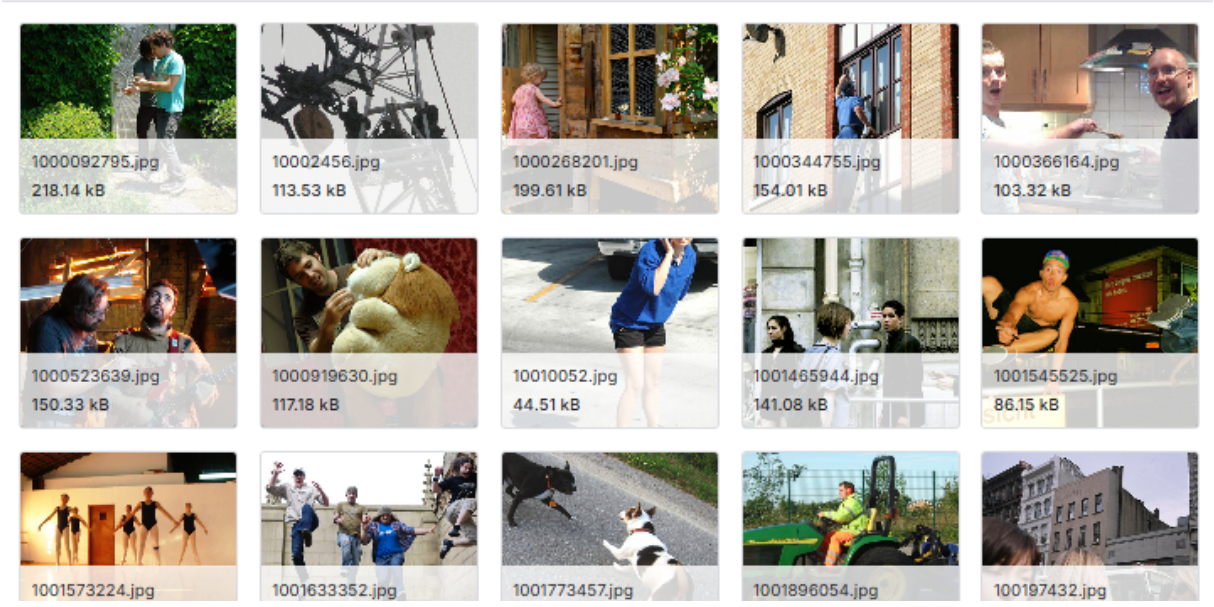

FIGURE 1 – Aperçu global du dataset Flickr30k. Chaque image possède plusieurs légendes.

2 Construction du vocabulaire

Pour représenter les mots des légendes, nous avons construit un vocabulaire à partir des annotations textuelles du dataset. Les étapes principales sont les suivantes :

1. Nettoyage des légendes : suppression des commentaires vides ou contenant uniquement des espaces.
2. Tokenisation des phrases à l'aide de la fonction `word_tokenize` de NLTK.
3. Comptage de la fréquence de chaque mot.
4. Filtrage des mots apparaissant moins de 2 fois dans le dataset.
5. Ajout de tokens spéciaux :

A man in a blue hard hat and orange safety vest stands in an intersection while holding a flag .

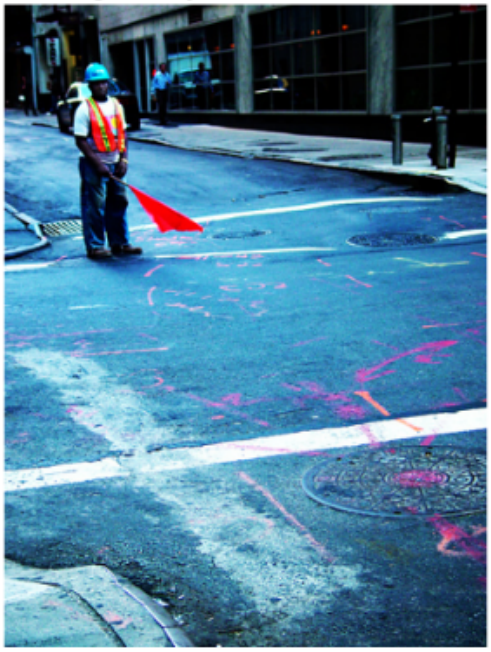

FIGURE 2 – Exemple d'une image du dataset avec sa légende.

— `<pad>` : token de remplissage pour les séquences courtes
— `<sos>` : début de séquence
— `<eos>` : fin de séquence
— `<unk>` : mots inconnus

Après ce filtrage, le vocabulaire contient **12 509 mots**. Les 10 mots les plus fréquents sont :

`a, ., in, the, on, and, man, is, of, with`

Ce vocabulaire sert à encoder les légendes en séquences d'indices pour l'entraînement du modèle.

3 Création du dataset PyTorch

Pour entraîner le modèle, nous avons créé un *dataset* personnalisé en héritant de la classe `Dataset` de PyTorch. La classe `Flickr30kDataset` effectue les opérations suivantes :

— Nettoyage du DataFrame pour supprimer les légendes vides.
— Groupement des légendes par image.
— Chargement des images depuis le dossier correspondant.
— Application des transformations définies pour l'augmentation et la normalisation des images.
— Sélection aléatoire d'une légende par image.
— Tokenisation, encodage et padding des légendes pour obtenir des séquences de longueur fixe (`max_length=30`).

3.1 Transformations appliquées aux images

Les transformations suivantes ont été utilisées pour préparer les images :
— Redimensionnement à 256×256 pixels
— Recadrage aléatoire à 224×224 pixels
— Flip horizontal aléatoire
— Conversion en tenseur PyTorch
— Normalisation avec les moyennes et écarts types standards d'ImageNet : $mean = [0.485, 0.456, 0.406]$, $std = [0.229, 0.224, 0.225]$

3.2 Séparation en ensembles d'entraînement et de test

Le dataset complet contient **31 783 images**. Il a été séparé en :
— Ensemble d'entraînement : 25 426 images
— Ensemble de test : 6 357 images

3.3 Exemple d'échantillon

Pour un exemple d'image et sa légende encodée :
— **Shape de l'image :** $3 \times 224 \times 224$ (canaux, hauteur, largeur)
— **Shape de la légende encodée :** 30 tokens
— **Premiers tokens de la légende encodée :** `[1, 4, 5, 6, 7, 8, 9, 10, 11, 12,...]`

Cette préparation garantit que chaque image et légende est prête pour l'entrée dans le modèle de captioning.

4 Création du dataset PyTorch

Pour entraîner le modèle, nous avons créé un *dataset* personnalisé en héritant de la classe `Dataset` de PyTorch. La classe `Flickr30kDataset` effectue les opérations suivantes :
— Nettoyage du DataFrame pour supprimer les légendes vides.
— Groupement des légendes par image.
— Chargement des images depuis le dossier correspondant.
— Application des transformations définies pour l'augmentation et la normalisation des images.
— Sélection aléatoire d'une légende par image.
— Tokenisation, encodage et padding des légendes pour obtenir des séquences de longueur fixe (`max_length=30`).

4.1 Transformations appliquées aux images

Les transformations suivantes ont été utilisées pour préparer les images :
— Redimensionnement à 256×256 pixels
— Recadrage aléatoire à 224×224 pixels
— Flip horizontal aléatoire
— Conversion en tenseur PyTorch
— Normalisation avec les moyennes et écarts types standards d'ImageNet : $mean = [0.485, 0.456, 0.406]$, $std = [0.229, 0.224, 0.225]$

4.2 Séparation en ensembles d'entraînement et de test

Le dataset complet contient **31 783 images**. Il a été séparé en :
— Ensemble d'entraînement : 25 426 images
— Ensemble de test : 6 357 images

4.3 Exemple d'échantillon

Pour un exemple d'image et sa légende encodée :
— **Shape de l'image :** $3 \times 224 \times 224$ (canaux, hauteur, largeur)
— **Shape de la légende encodée :** 30 tokens
— **Premiers tokens de la légende encodée :** `[1, 4, 5, 6, 7, 8, 9, 10, 11, 12,...]`

Cette préparation garantit que chaque image et légende est prête pour l'entrée dans le modèle de captioning.

4.4 Création des DataLoaders

Pour entraîner le modèle de manière efficace, nous avons utilisé les `DataLoader` de PyTorch, qui permettent de charger les données par batchs.

Une fonction personnalisée `collate_fn` a été définie pour :
— Empiler les images dans un tenseur de forme $(batch\_size, 3, 224, 224)$
— Empiler les légendes encodées dans un tenseur de forme $(batch\_size, max\_length)$

Les paramètres des DataLoaders sont les suivants :
— **Batch size :** 32
— **Train loader :** mélange aléatoire des données (`shuffle=True`)
— **Test loader :** pas de mélange (`shuffle=False`)
— **Nombre de workers :** 2

4.5 Exemple d'un batch

Pour un batch du DataLoader d'entraînement :
— **Shape des images :** $(32, 3, 224, 224)$
— **Shape des légendes :** $(32, 30)$
— **Première légende encodée (10 premiers tokens) :** `[1, 4, 29, 7, 1272, 17, 18, 655, 20, 2]`

Cette étape garantit que les images et les légendes sont correctement groupées et prêtes pour l'entraînement du modèle.

5 Couche d'embeddings

Pour représenter les mots sous forme de vecteurs numériques, nous avons implémenté une couche d'embeddings avec PyTorch. Cette couche permet de transformer chaque mot en un vecteur dense de dimension fixe (`embedding_dim`) pour l'entrée dans le modèle LSTM.

5.1 Implémentation

La classe `EmbeddingLayer` permet de :
— Utiliser des embeddings pré-entraînés (*Word2Vec*) si disponibles.
— Sinon, initialiser les embeddings aléatoirement avec la méthode Xavier.
— Convertir les indices des mots en vecteurs de dimension 300.

5.2 Paramètres de la couche

— **Taille du vocabulaire :** 12 509 mots
— **Dimension de l'embedding :** 300

Cette couche d'embeddings est utilisée en entrée du LSTM avec attention pour encoder les légendes textuelles.

6 Extracteur de caractéristiques avec ResNet50

Pour extraire des représentations visuelles des images, nous avons utilisé un ResNet50 pré-entraîné sur ImageNet. Ce modèle permet de transformer chaque image en un ensemble de vecteurs de caractéristiques riche en informations.

6.1 Implémentation

La classe `FeatureExtractor` effectue les opérations suivantes :
— Chargement du ResNet50 pré-entraîné.
— Conservation de toutes les couches sauf les deux dernières couches pleinement connectées et la couche d'average pooling finale.
— Gel des poids du ResNet afin qu'ils ne soient pas mis à jour lors de l'entraînement.
— Application d'un `AdaptiveAvgPool2d` pour obtenir une sortie de taille 7×7 par carte de caractéristiques.
— Réorganisation des dimensions pour obtenir un tenseur de forme $(batch, 49, 2048)$, où 49 correspond aux 7x7 positions spatiales et 2048 à la dimension des features.

6.2 Exemple de features extraites

Pour un batch de 2 images :
— **Shape des features :** $(2, 49, 2048)$
— **Nombre de features par image :** 49
— **Dimension de chaque feature :** 2048

Cette représentation est ensuite utilisée comme entrée du module d'attention pour guider la génération des légendes.

7 Implémentation du module d'attention

Le module d'attention constitue une composante essentielle du modèle d'Image Captioning. Il permet au décodeur LSTM de se concentrer dynamiquement sur les régions pertinentes de l'image à chaque étape de génération. Plutôt que de traiter toutes les informations visuelles de manière uniforme, le mécanisme d'attention attribue des poids

différents aux différentes régions, renforçant ainsi la capacité du modèle à produire des légendes cohérentes et pertinentes.

Le principe repose sur le calcul d'un vecteur de contexte à chaque pas temporel. Ce vecteur est obtenu en combinant les caractéristiques extraites par l'encodeur CNN (ici ResNet50) avec l'état caché actuel du décodeur. Des scores d'attention sont calculés pour chaque région de l'image, puis normalisés afin de produire des poids d'attention. Le vecteur de contexte est ensuite obtenu comme combinaison pondérée des caractéristiques visuelles, reflétant l'importance relative de chaque région pour la prédiction du mot suivant.

La validation du module d'attention a été réalisée en utilisant des caractéristiques et des états cachés simulés. Les résultats montrent que le vecteur de contexte produit pour chaque échantillon possède la dimension attendue et que les poids d'attention sont correctement normalisés, leur somme étant égale à 1. Ces observations confirment le bon fonctionnement du mécanisme et sa capacité à fournir des informations visuelles pertinentes pour le décodeur LSTM.

8 Décodeur LSTM avec mécanisme d'attention

Le décodeur repose sur un réseau LSTM enrichi par un mécanisme d'attention, permettant de générer une légende mot par mot à partir des caractéristiques visuelles extraites par l'encodeur CNN. À chaque étape temporelle, le décodeur ne se contente pas d'utiliser l'état caché précédent, mais s'appuie également sur un vecteur de contexte calculé par le module d'attention.

L'état caché initial du LSTM est obtenu à partir de la moyenne des caractéristiques visuelles de l'image. Cette stratégie permet d'injecter une information globale sur l'image dès le début du processus de génération, assurant une meilleure cohérence sémantique des premières prédictions.

À chaque pas de temps, le mécanisme d'attention calcule des poids d'importance sur les différentes régions de l'image, en fonction de l'état caché courant du décodeur. Ces poids sont utilisés pour produire un vecteur de contexte, représentant les zones les plus pertinentes de l'image pour la prédiction du mot suivant.

Le vecteur de contexte est concaténé avec l'embedding du mot courant et transmis à la cellule LSTM. L'état caché mis à jour est ensuite projeté dans l'espace du vocabulaire afin de prédire la distribution de probabilité du mot suivant. Un dropout est appliqué pour réduire le surapprentissage et améliorer la généralisation du modèle.

9 Architecture complète du modèle

Le modèle d'Image Captioning adopté dans ce travail repose sur une architecture encodeur-décodeur. L'encodeur est basé sur un réseau ResNet50 pré-entraîné, utilisé comme extracteur de caractéristiques visuelles. Les couches de classification finales sont supprimées afin d'obtenir une représentation spatiale riche de l'image sous forme de régions.

Les caractéristiques extraites par l'encodeur sont ensuite transmises au décodeur LSTM avec attention. Les légendes textuelles sont converties en vecteurs denses à l'aide d'une couche d'embedding, permettant de représenter les mots dans un espace sémantique continu. Ces embeddings sont utilisés comme entrée du décodeur pendant l'entraînement.

Lors de la phase d'inférence, la génération de la légende débute par un token de début de séquence. Le modèle prédit ensuite les mots successifs de manière itérative jusqu'à la génération du token de fin de séquence ou l'atteinte d'une longueur maximale prédéfinie. Cette approche permet de produire des légendes complètes et cohérentes, adaptées au contenu visuel de l'image.

10 Procédure d'entraînement

L'entraînement du modèle est réalisé sur un sous-ensemble du jeu de données, représentant 50 % des images disponibles pour l'apprentissage, tandis que les 50 % restants sont utilisés pour la validation. Cette stratégie permet d'évaluer la capacité de généralisation du modèle sur des données non vues.

La fonction de perte utilisée est l'entropie croisée catégorielle, en ignorant les tokens de remplissage afin de ne pas biaiser l'apprentissage. L'optimisation est effectuée à l'aide de l'algorithme Adam, reconnu pour sa stabilité et sa rapidité de convergence. Un scheduler de type *StepLR* est appliqué afin de réduire progressivement le taux d'apprentissage au cours de l'entraînement.

Afin d'éviter l'explosion des gradients, une technique de *gradient clipping* est employée. À intervalles réguliers, des exemples de légendes générées sont affichés afin de suivre qualitativement l'évolution des performances du modèle au fil des époques.

11 Analyse de la convergence

La figure illustre l'évolution de la fonction de perte au cours des différentes époques d'entraînement. On observe une diminution progressive de la perte d'entraînement, indiquant une amélioration continue des performances du modèle. La perte de validation suit une tendance similaire, suggérant une bonne capacité de généralisation et l'absence de surapprentissage significatif.

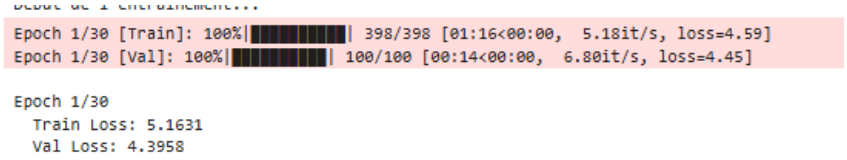

FIGURE 3 – epcph 1/30

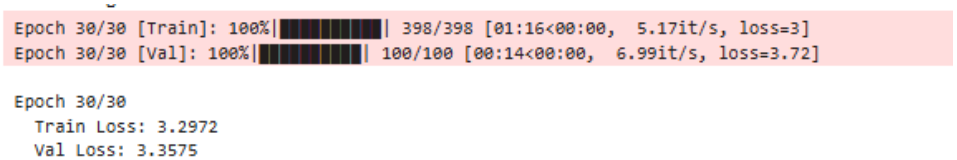

FIGURE 4 – epoch 30/30

12 Évaluation du modèle

L'évaluation du modèle a été réalisée sur un sous-ensemble du jeu de données de test, totalement distinct des données utilisées lors de l'entraînement. Cette étape vise à analyser la capacité de généralisation du modèle et à évaluer la qualité des légendes générées pour des images jamais vues auparavant.

L'évaluation repose à la fois sur des critères quantitatifs, à travers le calcul de la fonction de perte, et sur une analyse qualitative, basée sur l'inspection visuelle des légendes générées.

12.1 Protocole d'évaluation

Lors de cette phase, le modèle est évalué sans *teacher forcing*. À chaque pas de temps, le mot prédit par le modèle est réutilisé comme entrée pour la génération du mot suivant, ce qui correspond à un scénario d'inférence réaliste.

La fonction de perte utilisée est l'entropie croisée, calculée entre les légendes prédites et les légendes de référence, en ignorant le symbole de `padding` afin de ne pas biaiser l'évaluation.

12.2 Évaluation quantitative

La performance quantitative du modèle est mesurée à l'aide de la perte moyenne calculée sur 512 exemples du jeu de test. Le résultat obtenu est le suivant :

$$\textbf{Loss moyenne} = \textbf{3.4595}$$

Cette valeur indique que le modèle parvient à apprendre une correspondance globale entre le contenu visuel des images et leur description textuelle. Néanmoins, une perte de cet ordre suggère que certaines prédictions restent approximatives, notamment pour des descriptions complexes ou riches en détails.

12.3 Évaluation qualitative

Afin de compléter l'analyse quantitative, une évaluation qualitative a été menée en comparant les légendes générées par le modèle avec les légendes de référence du jeu de test. Plusieurs exemples représentatifs sont présentés ci-dessous.

- **Exemple 1 :** La légende générée décrit correctement la scène principale, en identifiant la présence d'un homme assis. Toutefois, certains détails visuels, tels que la tenue vestimentaire ou l'objet tenu, diffèrent de la description originale.
- **Exemple 2 :** Le modèle reconnaît correctement le sujet principal de l'image, mais échoue à capturer précisément l'action, remplaçant une scène de course par une scène de déplacement plus générique.
- **Exemple 3 :** La légende générée présente des répétitions excessives de mots, ce qui traduit une difficulté du modèle à maintenir une cohérence linguistique sur des séquences longues et à diversifier le vocabulaire utilisé.

12.4 Analyse et discussion

Les résultats obtenus montrent que le modèle est capable d'extraire des informations visuelles pertinentes et de produire des légendes globalement cohérentes. Le mécanisme d'attention permet au décodeur de se focaliser sur différentes régions de l'image au cours de la génération de la légende.

Cependant, certaines limites subsistent, notamment dans la précision des actions décrites et dans la gestion des répétitions lexicales. Ces limitations peuvent être attribuées à la taille du modèle, à la complexité du vocabulaire ou à la stratégie de décodage utilisée.

Des améliorations futures pourraient inclure l'utilisation de techniques de décodage avancées telles que le *beam search*, l'augmentation de la capacité du décodeur, ou encore l'intégration de métriques d'évaluation standard en génération de texte telles que BLEU, METEOR ou CIDEr.

Conclusion de l'évaluation

En conclusion, l'évaluation met en évidence que le modèle d'image captioning basé sur un extracteur ResNet et un décodeur LSTM avec mécanisme d'attention est capable de générer des descriptions pertinentes des images. Malgré certaines erreurs et approximations, les résultats obtenus sont encourageants et constituent une base solide pour des améliorations futures.

13 Visualisation du mécanisme d'attention

Afin de mieux comprendre le comportement interne du modèle et d'interpréter ses décisions, une visualisation du mécanisme d'attention a été réalisée lors de la génération des légendes. Cette analyse permet d'observer comment le modèle répartit son attention sur les différentes régions de l'image à chaque étape de génération des mots.

13.1 Principe de la visualisation

Lors de la génération d'une légende, le module d'attention produit, pour chaque mot généré, un ensemble de poids associés aux différentes régions spatiales de l'image. Dans notre cas, l'image est représentée par une grille de 7×7, soit 49 régions distinctes, issues de l'encodeur ResNet50.

Pour une légende de longueur T, le mécanisme d'attention génère une matrice de poids de dimension $(T, 49)$, où chaque ligne correspond à la distribution d'attention associée à un mot donné. Ces poids sont normalisés, leur somme étant égale à 1 pour chaque mot, ce qui permet une interprétation probabiliste.

13.2 Résultat observé

Pour l'exemple étudié, le modèle a généré la légende suivante :

"a man in a red shirt and a blue shirt is riding a bike in a park."

La visualisation des poids d'attention montre que :

— la matrice d'attention obtenue est de dimension $(19, 49)$, correspondant à une légende de 19 mots et 49 régions d'image ;

— pour les premiers mots de la légende, les poids d'attention sont relativement répartis sur plusieurs régions, traduisant une phase de compréhension globale de la scène ;
— à mesure que la génération progresse, l'attention se concentre davantage sur des zones spécifiques de l'image, notamment celles correspondant au sujet principal et à l'action décrite (l'homme et le vélo).

13.3 Analyse

Cette visualisation confirme que le mécanisme d'attention permet au modèle d'ajuster dynamiquement son focus spatial en fonction du mot à générer. Les mots décrivant les objets principaux et les actions reçoivent une attention plus marquée sur les régions pertinentes de l'image, tandis que les mots fonctionnels (articles, prépositions) sont associés à une attention plus diffuse.

Cependant, certaines incohérences dans la légende générée, comme la répétition ou la confusion entre les couleurs des vêtements, suggèrent que le modèle peut parfois répartir son attention sur des régions visuellement proches ou ambiguës.

Conclusion sur l'attention

La visualisation du mécanisme d'attention constitue un outil essentiel pour interpréter le fonctionnement du modèle. Elle démontre que le décodeur LSTM exploite efficacement les représentations spatiales fournies par l'encodeur, tout en mettant en évidence les limites actuelles du modèle en termes de précision descriptive.

14 Conclusion générale

Dans ce travail, nous avons implémenté et évalué un modèle complet d'Image Captioning combinant un encodeur convolutionnel pré-entraîné de type ResNet50 et un décodeur séquentiel basé sur un LSTM enrichi par un mécanisme d'attention. Cette architecture permet de relier efficacement les informations visuelles extraites des images aux structures séquentielles du langage naturel.

Les résultats expérimentaux montrent que le modèle est capable de générer des légendes globalement cohérentes et pertinentes, décrivant correctement les objets principaux et certaines actions présentes dans les images. L'utilisation du Transfer Learning a permis de tirer parti de représentations visuelles riches, tout en réduisant le coût d'entraînement.

L'évaluation quantitative, basée sur la fonction de perte, ainsi que l'analyse qualitative des légendes générées, mettent en évidence les forces du modèle, mais aussi certaines limites, notamment la tendance à produire des descriptions génériques ou des répétitions lexicales pour des scènes complexes.

La visualisation du mécanisme d'attention a permis d'interpréter le comportement interne du modèle et de confirmer sa capacité à se focaliser dynamiquement sur différentes régions de l'image en fonction du mot à générer. Cette interprétabilité constitue un atout majeur des modèles avec attention par rapport aux architectures séquentielles classiques.

En perspective, plusieurs améliorations peuvent être envisagées, telles que l'utilisation de stratégies de décodage avancées (beam search), l'intégration de métriques d'évaluation standardisées pour l'image captioning (BLEU, METEOR, CIDEr), ou encore l'adoption de modèles plus récents basés sur les Transformers. Néanmoins, les résultats obtenus dans ce projet constituent une base solide pour la compréhension et la mise en œuvre des systèmes modernes de génération automatique de descriptions d'images.