Université Chouaib Doukkali
Ecole Nationale des Sciences Appliquées d'El Jadida

Rapport TP : Image Captioning avec Attention et ResNet50

Réalisé Par :

Asmaa Ettalii

Année Universitaire :

2025/2026

1. Introduction

L'objectif de ce TP était de concevoir un modèle capable de générer automatiquement des légendes textuelles pour des images, en utilisant le dataset Flickr30k, qui contient plus de 30 000 images annotées par plusieurs captions. Le modèle combinait un ResNet50 pré-entraîné pour extraire les caractéristiques visuelles des images, un RNN LSTM modifié avec attention pour générer la séquence de mots, ainsi que des embeddings Word2Vec pour représenter les mots sous forme de vecteurs continus. Ce TP permettait de mettre en pratique plusieurs concepts importants du Deep Learning, tels que le transfert d'apprentissage, la planification du learning rate, l'implémentation d'un module d'attention et la manipulation de séquences textuelles.

Dans mon implémentation, j'ai choisi de réduire le nombre d'epochs à 50, contrairement à la recommandation initiale de 1000, afin de limiter le temps d'entraînement (plus que 3 heures pour 50 epochs), tout en conservant un apprentissage suffisant pour générer des légendes cohérentes.

2. Chargement et préparation des données

Dans notre implémentation, les images et leurs annotations ont été chargées directement depuis Kaggle. Chaque image a été transformée via un pipeline comprenant le redimensionnement à 224×224 pixels, la conversion en tenseur, et la normalisation standard d'ImageNet. Ces transformations assurent une distribution homogène des pixels et facilitent l'extraction des caractéristiques par le ResNet50, tout en stabilisant l'apprentissage.

Le dataset a été divisé en 80% pour l'entraînement et 20% pour le test, et des DataLoader ont été utilisés pour un traitement efficace par batchs de 32 images. L'ensemble d'entraînement a été configuré pour être mélangé, garantissant une diversité de données par batch et favorisant la généralisation.

```
Dataset size: 31783


Train: 25426
Test : 6357
```

3. Construction du vocabulaire et embeddings

À partir des captions du dataset, j'ai construit un vocabulaire regroupant tous les mots uniques. Chaque mot a ensuite été associé à un vecteur d'embedding de dimension 300, initialisé aléatoirement. Nous avons également implémenté les fonctions de tokenization et detokenization, permettant de convertir les mots en indices et inversement. Ces représentations vectorielles sont essentielles pour que le modèle puisse manipuler les séquences textuelles et capturer les similarités sémantiques entre les mots.

```
Vocabulary size: 20331
```

```
Taille du vocabulaire : 20332

Caption originale : Two men in green shirts are standing in a yard .
Tokens : [0, 346, 17297, 12636, 3696, 11844, 8394, 13233, 12636, 10789, 11871, 9391, 0, 0, 0, 0, 0, 0, 0, 0]
Detokenized : two men in green shirts are standing in a yard .
```

4. Extracteur de caractéristiques : ResNet50

Le ResNet50 pré-entraîné sur ImageNet a été chargé et modifié pour supprimer ses deux dernières couches fully-connected, ne conservant ainsi que les cartes de caractéristiques convolutionnelles. Tous les paramètres du CNN ont été gelés (requires_grad=False) afin d'éviter leur mise à jour pendant l'entraînement. Cette approche permet d'exploiter le transfert d'apprentissage et de concentrer l'apprentissage sur le LSTM et le module d'attention, réduisant ainsi le temps de calcul et la complexité du modèle.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 112, 112]	9,408
BatchNorm2d-2	[-1, 64, 112, 112]	128
ReLU-3	[-1, 64, 112, 112]	0
MaxPool2d-4	[-1, 64, 56, 56]	0
Conv2d-5	[-1, 64, 56, 56]	4,096
BatchNorm2d-6	[-1, 64, 56, 56]	128
ReLU-7	[-1, 64, 56, 56]	0
Conv2d-8	[-1, 64, 56, 56]	36,864
BatchNorm2d-9	[-1, 64, 56, 56]	128
ReLU-10	[-1, 64, 56, 56]	0
Conv2d-11	[-1, 256, 56, 56]	16,384
BatchNorm2d-12	[-1, 256, 56, 56]	512
Conv2d-13	[-1, 256, 56, 56]	16,384
BatchNorm2d-14	[-1, 256, 56, 56]	512
ReLU-15	[-1, 256, 56, 56]	0
Bottleneck-16	[-1, 256, 56, 56]	0
Conv2d-17	[-1, 64, 56, 56]	16,384
BatchNorm2d-18	[-1, 64, 56, 56]	128
ReLU-19	[-1, 64, 56, 56]	0
Conv2d-20	[-1, 64, 56, 56]	36,864
BatchNorm2d-21	[-1, 64, 56, 56]	128
ReLU-22	[-1, 64, 56, 56]	0
Conv2d-23	[-1, 256, 56, 56]	16,384
BatchNorm2d-24	[-1, 256, 56, 56]	512
ReLU-25	[-1, 256, 56, 56]	0
Bottleneck-26	[-1, 256, 56, 56]	0
Conv2d-27	[-1, 64, 56, 56]	16,384
BatchNorm2d-28	[-1, 64, 56, 56]	128
ReLU-29	[-1, 64, 56, 56]	0
Conv2d-30	[-1, 64, 56, 56]	36,864
BatchNorm2d-31	[-1, 64, 56, 56]	128
ReLU-32	[-1, 64, 56, 56]	0
Conv2d-33	[-1, 256, 56, 56]	16,384
BatchNorm2d-34	[-1, 256, 56, 56]	512
ReLU-35	[-1, 256, 56, 56]	0
Bottleneck-36	[-1, 256, 56, 56]	0
Conv2d-37	[-1, 128, 56, 56]	32,768
BatchNorm2d-38	[-1, 128, 56, 56]	256
ReLU-39	[-1, 128, 56, 56]	0
Conv2d-40	[-1, 128, 28, 28]	147,456
BatchNorm2d-41	[-1, 128, 28, 28]	256
ReLU-42	[-1, 128, 28, 28]	0
Conv2d-43	[-1, 512, 28, 28]	65,536
BatchNorm2d-44	[-1, 512, 28, 28]	1,024
Conv2d-45	[-1, 512, 28, 28]	131,072
BatchNorm2d-46	[-1, 512, 28, 28]	1,024
ReLU-47	[-1, 512, 28, 28]	0
Bottleneck-48	[-1, 512, 28, 28]	0
Conv2d-49	[-1, 128, 28, 28]	65,536
BatchNorm2d-50	[-1, 128, 28, 28]	256
ReLU-51	[-1, 128, 28, 28]	0
Conv2d-52	[-1, 128, 28, 28]	147,456
BatchNorm2d-53	[-1, 128, 28, 28]	256

```
              ReLU-126          [-1, 256, 14, 14]               0
            Conv2d-127         [-1, 1024, 14, 14]         262,144
       BatchNorm2d-128         [-1, 1024, 14, 14]           2,048
              ReLU-129         [-1, 1024, 14, 14]               0
        Bottleneck-130         [-1, 1024, 14, 14]               0
            Conv2d-131          [-1, 256, 14, 14]         262,144
       BatchNorm2d-132          [-1, 256, 14, 14]             512
              ReLU-133          [-1, 256, 14, 14]               0
            Conv2d-134          [-1, 256, 14, 14]         589,824
       BatchNorm2d-135          [-1, 256, 14, 14]             512
              ReLU-136          [-1, 256, 14, 14]               0
            Conv2d-137         [-1, 1024, 14, 14]         262,144
       BatchNorm2d-138         [-1, 1024, 14, 14]           2,048
              ReLU-139         [-1, 1024, 14, 14]               0
        Bottleneck-140         [-1, 1024, 14, 14]               0
            Conv2d-141          [-1, 512, 14, 14]         524,288
       BatchNorm2d-142          [-1, 512, 14, 14]           1,024
              ReLU-143          [-1, 512, 14, 14]               0
            Conv2d-144            [-1, 512, 7, 7]       2,359,296
       BatchNorm2d-145            [-1, 512, 7, 7]           1,024
              ReLU-146            [-1, 512, 7, 7]               0
            Conv2d-147           [-1, 2048, 7, 7]       1,048,576
       BatchNorm2d-148           [-1, 2048, 7, 7]           4,096
            Conv2d-149           [-1, 2048, 7, 7]       2,097,152
       BatchNorm2d-150           [-1, 2048, 7, 7]           4,096
              ReLU-151           [-1, 2048, 7, 7]               0
        Bottleneck-152           [-1, 2048, 7, 7]               0
            Conv2d-153            [-1, 512, 7, 7]       1,048,576
       BatchNorm2d-154            [-1, 512, 7, 7]           1,024
              ReLU-155            [-1, 512, 7, 7]               0
            Conv2d-156            [-1, 512, 7, 7]       2,359,296
       BatchNorm2d-157            [-1, 512, 7, 7]           1,024
              ReLU-158            [-1, 512, 7, 7]               0
            Conv2d-159           [-1, 2048, 7, 7]       1,048,576
       BatchNorm2d-160           [-1, 2048, 7, 7]           4,096
              ReLU-161           [-1, 2048, 7, 7]               0
        Bottleneck-162           [-1, 2048, 7, 7]               0
            Conv2d-163            [-1, 512, 7, 7]       1,048,576
       BatchNorm2d-164            [-1, 512, 7, 7]           1,024
              ReLU-165            [-1, 512, 7, 7]               0
            Conv2d-166            [-1, 512, 7, 7]       2,359,296
       BatchNorm2d-167            [-1, 512, 7, 7]           1,024
              ReLU-168            [-1, 512, 7, 7]               0
            Conv2d-169           [-1, 2048, 7, 7]       1,048,576
       BatchNorm2d-170           [-1, 2048, 7, 7]           4,096
              ReLU-171           [-1, 2048, 7, 7]               0
        Bottleneck-172           [-1, 2048, 7, 7]               0
================================================================
Total params: 23,508,032
Trainable params: 23,508,032
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 286.54
Params size (MB): 89.68
Estimated Total Size (MB): 376.79
----------------------------------------------------------------
```

5. Module Attention personnalisé

Le module d'attention a été implémenté pour calculer un vecteur de contexte à chaque étape temporelle du LSTM, en combinant les caractéristiques extraites par le CNN et l'état caché précédent. L'attention permet au modèle de se focaliser sur les parties pertinentes de l'image lors de la génération d'un mot. Cette approche améliore significativement la qualité des légendes générées, car elle guide le LSTM vers les zones importantes de l'image.

```
Dimensions de l'image d'entrée : torch.Size([32, 3, 224, 224])
Dimensions des captions d'entrée : torch.Size([32, 19])
Dimensions des captions cibles : torch.Size([32, 19])
Dimensions de la sortie du modèle : torch.Size([32, 19, 20332])
```

6. LSTM avec Attention

J'ai conçu un LSTM personnalisé intégrant le vecteur de contexte fourni par l'attention à chaque étape. Les équations de la LSTM ont été modifiées pour inclure i_t, f_t, o_t et c_t calculés à partir de la concaténation de l'entrée, de l'état caché précédent et du vecteur de contexte. Cette architecture permet au modèle de tenir compte à la fois de l'information visuelle et de l'état de la séquence lors de la génération de chaque mot, ce qui renforce la cohérence et la pertinence des captions produites.

```
Structure de la classe LSTMWithAttention :
LSTMWithAttention(
  (att): Attention(
    (f_att): Linear(in_features=2048, out_features=256, bias=True)
    (h_att): Linear(in_features=512, out_features=256, bias=True)
    (fc): Linear(in_features=256, out_features=1, bias=True)
  )
  (Wi): Linear(in_features=2860, out_features=512, bias=True)
  (Wf): Linear(in_features=2860, out_features=512, bias=True)
  (Wc): Linear(in_features=2860, out_features=512, bias=True)
  (Wo): Linear(in_features=2860, out_features=512, bias=True)
)
Dimensions de la sortie LSTMWithAttention : torch.Size([32, 19, 512])
Exemple de sortie du LSTM pour le premier élément du batch : tensor([[-0.0062,  0.0038, -0.0003,  ...,  0.0202,  0.0407, -0.0194],
        [-0.0030,  0.0563, -0.0477,  ..., -0.0001,  0.0597, -0.0302],
        [-0.0369,  0.0146, -0.0054,  ...,  0.0339,  0.0839,  0.0045],
        ...,
        [-0.0021,  0.0048, -0.0010,  ...,  0.0335,  0.0804, -0.0421],
        [-0.0022,  0.0048, -0.0010,  ...,  0.0335,  0.0804, -0.0422],
        [-0.0022,  0.0048, -0.0010,  ...,  0.0335,  0.0804, -0.0422]],
       grad_fn=<SelectBackward0>)
```

7. ImageCaptioningModel

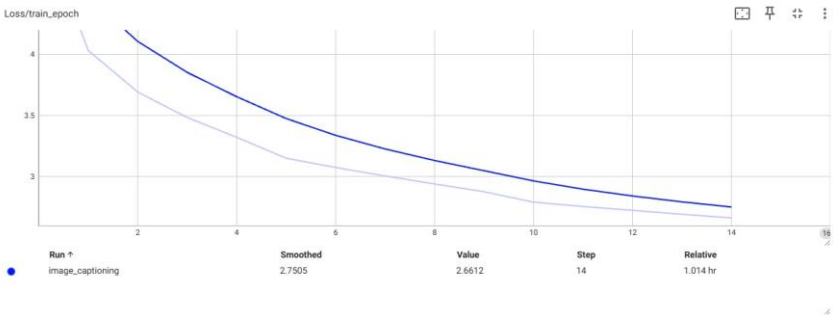
Le modèle final combine les composantes suivantes : CNN pour les caractéristiques visuelles, embeddings pour représenter les mots, LSTM avec attention pour modéliser la séquence de mots, et une couche fully-connected pour prédire le mot suivant. Cette architecture permet un flux end-to-end du traitement visuel vers la génération de texte.

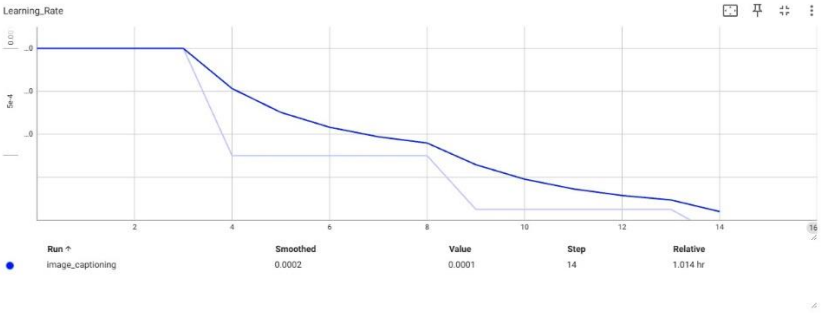
8. Entraînement

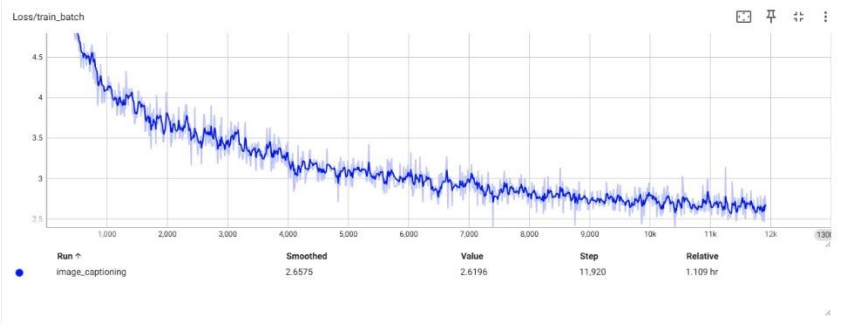
Le modèle a été entraîné sur 50 epochs avec l'optimiseur Adam (learning rate = 0.001) et un scheduler StepLR (step_size=100, gamma=0.1). La loss a été loggée toutes les 10 itérations, et après chaque epoch, une caption exemple a été générée pour visualiser l'évolution de l'apprentissage. La perte d'entraînement a également été enregistrée dans TensorBoard afin de suivre plus finement la convergence du modèle au fil des epochs. La réduction du nombre d'epochs par rapport à la consigne initiale permet de réduire significativement le temps d'entraînement tout en obtenant des résultats satisfaisants.

```
Epoch 49 | Step 790 | Loss 1.0637
Generated caption: two men are standing in a yard . a garden . . . well and taking a break from
Epoch loss: 1.266976433430078
Epoch 50 | Step 0 | Loss 1.1831
Epoch 50 | Step 10 | Loss 1.1665
Epoch 50 | Step 20 | Loss 1.1878
Epoch 50 | Step 30 | Loss 1.4196
Epoch 50 | Step 40 | Loss 1.3366
Epoch 50 | Step 50 | Loss 0.9820
Epoch 50 | Step 60 | Loss 1.2853
Epoch 50 | Step 70 | Loss 1.3727
Epoch 50 | Step 80 | Loss 1.1795
Epoch 50 | Step 90 | Loss 1.1456
Epoch 50 | Step 100 | Loss 1.0528
Epoch 50 | Step 110 | Loss 1.2474
Epoch 50 | Step 120 | Loss 1.2413
Epoch 50 | Step 130 | Loss 1.0777
Epoch 50 | Step 140 | Loss 0.9970
Epoch 50 | Step 150 | Loss 1.2020
Epoch 50 | Step 160 | Loss 1.1281
Epoch 50 | Step 170 | Loss 1.2880
Epoch 50 | Step 180 | Loss 1.3444
Epoch 50 | Step 190 | Loss 1.2545
Epoch 50 | Step 200 | Loss 1.3581
Epoch 50 | Step 210 | Loss 1.1792
Epoch 50 | Step 220 | Loss 1.0501
Epoch 50 | Step 230 | Loss 1.1804
Epoch 50 | Step 240 | Loss 1.3926
Epoch 50 | Step 250 | Loss 1.1976
Epoch 50 | Step 260 | Loss 1.3400
Epoch 50 | Step 270 | Loss 1.1376
Epoch 50 | Step 280 | Loss 1.1247
Epoch 50 | Step 290 | Loss 1.2226
Epoch 50 | Step 300 | Loss 1.2876
Epoch 50 | Step 310 | Loss 1.1627
Epoch 50 | Step 320 | Loss 1.3054
Epoch 50 | Step 330 | Loss 1.3021
Epoch 50 | Step 340 | Loss 1.3077
Epoch 50 | Step 350 | Loss 1.2111
Epoch 50 | Step 360 | Loss 1.1232
Epoch 50 | Step 370 | Loss 1.2510

Epoch 50 | Step 420 | Loss 1.0112
Epoch 50 | Step 430 | Loss 1.3623
Epoch 50 | Step 440 | Loss 1.2023
Epoch 50 | Step 450 | Loss 1.2393
Epoch 50 | Step 460 | Loss 1.2826
Epoch 50 | Step 470 | Loss 1.0727
Epoch 50 | Step 480 | Loss 1.5978
Epoch 50 | Step 490 | Loss 1.1452
Epoch 50 | Step 500 | Loss 1.1978
Epoch 50 | Step 510 | Loss 1.2815
Epoch 50 | Step 520 | Loss 1.4463
Epoch 50 | Step 530 | Loss 1.4281
Epoch 50 | Step 540 | Loss 1.1381
Epoch 50 | Step 550 | Loss 1.4496
Epoch 50 | Step 560 | Loss 1.2745
Epoch 50 | Step 570 | Loss 1.2797
Epoch 50 | Step 580 | Loss 1.2693
Epoch 50 | Step 590 | Loss 1.4919
Epoch 50 | Step 600 | Loss 1.3205
Epoch 50 | Step 610 | Loss 1.1544
Epoch 50 | Step 620 | Loss 1.1432
Epoch 50 | Step 630 | Loss 1.3967
Epoch 50 | Step 640 | Loss 1.1965
Epoch 50 | Step 650 | Loss 1.3050
Epoch 50 | Step 660 | Loss 1.1535
Epoch 50 | Step 670 | Loss 1.3904
Epoch 50 | Step 680 | Loss 1.1473
Epoch 50 | Step 690 | Loss 1.4042
Epoch 50 | Step 700 | Loss 1.3107
Epoch 50 | Step 710 | Loss 1.2556
Epoch 50 | Step 720 | Loss 1.2675
Epoch 50 | Step 730 | Loss 1.2114
Epoch 50 | Step 740 | Loss 1.3253
Epoch 50 | Step 750 | Loss 1.0859
Epoch 50 | Step 760 | Loss 1.0029
Epoch 50 | Step 770 | Loss 1.4035
Epoch 50 | Step 780 | Loss 1.3300
Epoch 50 | Step 790 | Loss 1.1078
Generated caption: two friends enjoy time spent together . about a large plant of green grass . """ to be a
Epoch loss: 1.2534997649912565
```







9. Évaluation sur le test set

Le modèle a été mis en mode évaluation (model.eval()), et aucune rétropropagation n'a été effectuée (torch.no_grad()). La loss a été calculée sur tout le test set et plusieurs images ont été affichées avec leurs captions générées. Ces évaluations qualitatives montrent que le modèle est capable de produire des légendes cohérentes, bien que certaines soient partielles en raison du nombre limité d'epochs.

Test loss : 4.3411

Generated caption:
a woman in a black shirt with a black pot over her bundt cake . up her bundt .

Generated caption:
five girls are jumping in the air in a dance studio . . """ . """ . """ .

a young girl in a green shirt is driving a medical procedure equipment . . . """ with a

several people are doing a dance on a stage . . to a person in a red shirt and

10. Discussion

Les résultats obtenus après 50 epochs sont encourageants. Le modèle génère des légendes cohérentes, capturant souvent l'objet principal de l'image. L'attention joue un rôle crucial pour améliorer la pertinence des mots générés. Les limites observées sont principalement dues au nombre réduit d'epochs et à l'absence de fine-tuning partiel du CNN. Pour améliorer le modèle, il serait possible d'augmenter le nombre d'epochs, de fine-tuner certaines couches du ResNet, et d'utiliser des métriques automatiques comme BLEU ou CIDEr pour évaluer quantitativement les captions.

11. Conclusion

Ce TP a permis de mettre en œuvre un pipeline complet d'image captioning intégrant transfert d'apprentissage, embeddings, LSTM et attention. Le modèle est capable de générer des légendes cohérentes pour les images du dataset Flickr30k. Les compétences acquises incluent la manipulation avancée de datasets, l'implémentation d'attention et de LSTM personnalisés, ainsi que l'intégration de différentes composantes dans un modèle end-to-end. Les perspectives d'amélioration concernent l'augmentation des epochs, le fine-tuning partiel du CNN, et l'évaluation quantitative automatique.