# Rapport de TP 2 : Attention

## Image Captioning avec RNN et Attention sur ResNet

**Étudiante :** SALMA LIDAME
**Classe :** Master SDIA
**Module :** IA générative et ingénierie des prompts

# 1. Introduction

### 1.1 Contexte

Le captioning d'images est une tâche fondamentale en vision par ordinateur qui consiste à générer automatiquement des descriptions textuelles pertinentes à partir d'images. Cette tâche combine à la fois la compréhension visuelle et la génération de langage naturel, faisant d'elle un problème particulièrement intéressant dans le domaine de l'intelligence artificielle.

### 1.2 Objectifs du TP

Ce travail pratique vise à implémenter un système complet d'image captioning en utilisant une architecture moderne combinant :

- **Transfer Learning** avec un ResNet50 pré-entraîné pour l'extraction de caractéristiques visuelles
- **Mécanisme d'Attention** pour permettre au modèle de se concentrer sur les parties pertinentes de l'image
- **LSTM personnalisé** intégrant l'attention pour la génération de séquences textuelles
- **Embeddings Word2Vec** pré-entraînés pour une meilleure représentation sémantique

### 1.3 Dataset

Le dataset **Flickr30k** qui contient plus de 31 000 images avec des descriptions textuelles associées. Plus précisément :

- **158 915 captions** au total (environ 5 captions par image)
- Chaque image possède plusieurs descriptions décrivant son contenu sous différents angles
- Division : **80% entraînement** (25 426 images) et **20% test** (6 357 images)

| | image_name | comment_number | comment |
|---|---|---|---|
| 0 | 1000092795.jpg | 0 | Two young guys with shaggy hair look at their ... |
| 1 | 1000092795.jpg | 1 | Two young , White males are outside near many ... |
| 2 | 1000092795.jpg | 2 | Two men in green shirts are standing in a yard . |
| 3 | 1000092795.jpg | 3 | A man in a blue shirt standing in a garden . |
| 4 | 1000092795.jpg | 4 | Two friends enjoy time spent together . |
| ... | ... | ... | ... |
| 158910 | 998845445.jpg | 0 | A man in shorts and a Hawaiian shirt leans ove... |
| 158911 | 998845445.jpg | 1 | A young man hanging over the side of a boat , ... |
| 158912 | 998845445.jpg | 2 | A man is leaning off of the side of a blue and... |
| 158913 | 998845445.jpg | 3 | A man riding a small boat in a harbor , with f... |
| 158914 | 998845445.jpg | 4 | A man on a moored blue and white boat with hil... |

158914 rows × 3 columns

# 2. Méthodologie et Architecture

## 2.1 Architecture Globale

Notre modèle suit une architecture **encoder-decoder avec attention** :

1. **Encoder (ResNet50)** : Extrait des features visuelles de dimension (2048, 7, 7)
    - Utilise un ResNet50 pré-entraîné sur ImageNet
    - Retire les couches de pooling final et de classification
    - Conserve uniquement les couches convolutionnelles pour obtenir une carte de features spatiale
2. **Mécanisme d'Attention** : Calcule des poids d'attention sur les features spatiales
    - Permet au modèle de "regarder" différentes régions de l'image à chaque étape
    - Dimension d'attention : 512
3. **Decoder (LSTM avec Attention)** : Génère la séquence de mots en tenant compte du contexte visuel
    - État caché LSTM : 512 dimensions
    - Intègre le vecteur de contexte d'attention à chaque pas de temps

## 2.2 Principe du Mécanisme d'Attention

Le mécanisme d'attention permet au modèle de se concentrer dynamiquement sur différentes régions de l'image lors de la génération de chaque mot. Le processus se déroule comme suit :

1. **Projection des features** : Les features visuelles (2048 dimensions) et l'état caché du LSTM (512 dimensions) sont projetés dans un espace d'attention commun (512 dimensions)

**Calcul des scores d'attention** :

*attention_scores = W_att * concat(features, hidden_state)*

2. **Normalisation via Softmax** : Les scores sont normalisés pour obtenir des poids d'attention qui somment à 1

**Vecteur de contexte** :

*context_vector = sum(softmax(attention_scores) * features)*

### 2.3 LSTM avec Attention

**Porte d'entrée :**

```
i_t = σ(W_i * concat(x_t, h_{t-1}, att_t))
```

**Porte d'oubli :**

```
f_t = σ(W_f * concat(x_t, h_{t-1}, att_t))
```

**État de cellule :**

```
c_t = f_t * c_{t-1} + i_t * tanh(W_c * concat(x_t, h_{t-1}, att_t))
```

**Porte de sortie :**

```
o_t = σ(W_o * concat(x_t, h_{t-1}, att_t))
```

**État caché :**

```
h_t = o_t * tanh(c_t)
```

# 3. Implémentation

## 3.1 Chargement et Prétraitement des Données

### 3.1.1 Chargement du Dataset Flickr30k

Le dataset est chargé à partir d'un fichier CSV contenant trois colonnes :

- `image_name` : Nom du fichier image
- `comment_number` : Numéro de la caption (0-4)
- `comment` : Texte de la description

Après nettoyage (suppression des valeurs nulles et des espaces), nous obtenons **158 914 captions valides**.

### 3.1.2 Construction du Vocabulaire

Le vocabulaire est construit en extrayant tous les mots uniques des captions :

- **20 273 mots** au total
- Ajout de **4 tokens spéciaux** :
  - `<PAD>` (index 0) : Pour le padding des séquences
  - `<START>` (index 1) : Début de séquence
  - `<END>` (index 2) : Fin de séquence
  - `<UNK>` (index 3) : Mots inconnus

Traitement du texte :

- Conversion en minuscules
- Suppression de la ponctuation (.,!?;:"()[])
- Tokenisation simple par espaces

### 3.1.3 Chargement des Embeddings Word2Vec

Utilisation des embeddings **Google News Word2Vec** pré-entraînés :

- **3 millions de mots** dans le modèle complet
- **300 dimensions** par vecteur
- **16 833 mots** de notre vocabulaire trouvés dans Word2Vec
- Les mots non trouvés et les tokens spéciaux reçoivent des vecteurs aléatoires
- **Embeddings gelés** (non entraînables) pour préserver les représentations sémantiques apprises

### 3.1.4 Tokenisation et Préparation des Séquences

La fonction de tokenisation traite chaque caption :

1. Ajout du token `<START>` au début
2. Conversion des mots en indices via le vocabulaire
3. Remplacement des mots inconnus par `<UNK>`
4. Ajout du token `<END>` à la fin

5. **Longueur maximale** : 20 tokens
6. **Padding** : Complétion avec <PAD> pour atteindre la longueur fixe

#### 3.1.5 DataLoader et Traitement par Batch

Un DataLoader personnalisé gère le chargement efficace :

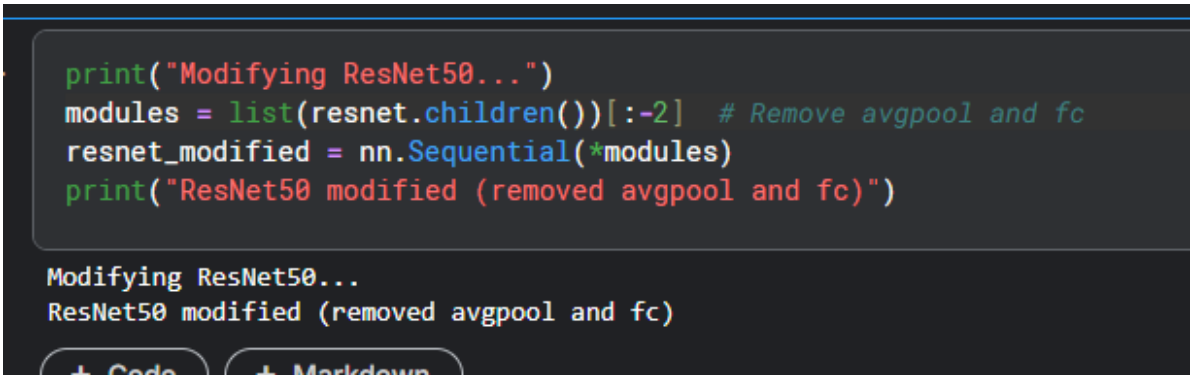
- **Batch size** : 32 images
- **Sélection aléatoire** d'une caption parmi les 5 disponibles pour chaque image
- **Collate function** : Empile les images et tokenise les captions de manière uniforme
- **Parallel loading** : 2 workers pour accélérer le chargement
- **Pin memory** : Optimisation pour le transfert CPU→GPU

### 3.2 Modèle ResNet50 pour Extraction de Features

#### Architecture

ResNet50 est utilisé comme extracteur de features visuelles :

- Chargement des **poids pré-entraînés sur ImageNet** (1,4 million d'images, 1000 classes)
- **Modification de l'architecture** : Suppression de la couche de Average Pooling et de la couche fully-connected finale
- **Sortie** : Carte de features de dimension (batch_size, 2048, 7, 7)
    - 2048 canaux (features)
    - 7×7 positions spatiales (49 régions de l'image)




#### Pourquoi geler les poids de ResNet ? (freeze parameters)

Le gel des paramètres de ResNet50 est crucial pour plusieurs raisons :

1. **Préservation des features** : ResNet50 a été entraîné sur ImageNet avec des millions d'images. Ces poids capturent des représentations visuelles génériques très utiles (contours, textures, formes, objets).

2. **Efficacité computationnelle** : En gelant les poids, nous réduisons considérablement le nombre de paramètres à optimiser.

## 3.3 Module d'Attention

Le module d'attention calcule quelles régions de l'image sont pertinentes à chaque pas de temps :

**Composants** :

- **Projection des features** : Linear(2048 → 512)
- **Projection de l'état caché** : Linear(512 → 512)
- **Couche d'attention** : Linear(512 → 1)
- **Activation** : ReLU sur la somme des projections
- **Normalisation** : Softmax sur les 49 positions spatiales

**Processus** :

1. Les features visuelles (2048, 7, 7) sont aplaties en (49, 2048)
2. Projection des features et de l'état caché dans l'espace d'attention
3. Calcul des scores d'attention pour chaque position (49 scores)
4. Application du softmax pour obtenir une distribution de probabilité
5. Calcul du vecteur de contexte comme moyenne pondérée des features

## 3.4 LSTM avec Attention

Notre implémentation LSTM intègre l'attention directement dans le calcul des portes :

**Architecture** :

- **Entrée combinée** : Embedding (300) + État caché (512) + Contexte d'attention (2048) = **2860 dimensions**
- **Quatre portes LSTM** :
    - Input gate : Contrôle quelles nouvelles informations sont ajoutées
    - Forget gate : Contrôle quelles informations sont oubliées
    - Cell gate : Génère les nouvelles informations candidates
    - Output gate : Contrôle la sortie basée sur la cell state
- **Dropout** : 50% appliqué sur la sortie pour régularisation

**Fonctionnement** :

- À chaque pas de temps, le contexte d'attention est recalculé en fonction de l'état caché actuel
- Ce contexte dynamique permet au modèle de "regarder" différentes parties de l'image pour chaque mot

## 3.5 Modèle Complet d'Image Captioning

Le modèle final intègre tous les composants :

**Structure** :

1. **Encoder CNN** : ResNet50 modifié (gelé)
2. **Embedding Layer** : Word2Vec (300 dim, gelé)
3. **Module d'Attention** : Calcul du contexte visuel
4. **LSTM avec Attention** : Génération de séquence
5. **Couche de sortie** : Linear(512 → 20273) pour prédire le vocabulaire

**Nombre de paramètres** :

- Total : ~14 millions de paramètres
- Entraînables : ~8 millions (sans ResNet et embeddings)

## 3.6 Entraînement

### Configuration

- **Fonction de perte** : CrossEntropyLoss avec ignore_index=0 (ignore les tokens `<PAD>`)
- **Optimiseur** : Adam avec learning rate initial de 0.001 et weight decay de 1e-5
- **Scheduler** : StepLR réduisant le learning rate de moitié tous les 10 epochs
- **Epochs** : 50 (avec early stopping à patience=5)
- **Gradient clipping** : Norm maximale de 5.0 pour éviter l'explosion des gradients
- **Teacher forcing** : Utilisation des mots réels pendant l'entraînement

### Processus d'entraînement

Pour chaque batch :

1. Extraction des features visuelles (sans gradients)
2. Embedding des captions (décalées d'un token pour teacher forcing)
3. Passage dans le LSTM avec attention
4. Prédiction du mot suivant
5. Calcul de la perte (ignoring padding)
6. Backpropagation avec clipping
7. Mise à jour des poids

### Monitoring

- **TensorBoard** : Visualisation de la loss en temps réel
- **Génération d'exemples** : À chaque epoch, génération d'une caption sur une image test
- **Sauvegarde** : Meilleur modèle basé sur la test loss
- **Checkpoints** : Sauvegarde tous les 5 epochs

### Génération de Captions (Inférence)

Le processus de génération est **autorégressif** :

1. Initialisation avec le token `<START>`
2. À chaque itération :

    - Embedding du token actuel
    - Calcul de l'attention sur l'image
    - Passage dans le LSTM
    - Prédiction du token suivant (argmax)
    - Arrêt si token `<END>` ou longueur maximale atteinte
3. Conversion des indices en mots

**Paramètre de température** : Peut être ajusté pour contrôler la créativité (température > 1 = plus aléatoire)

# 4. Résultats et Analyse

## 4.1 Évolution de la Loss

Durant les **50 epochs d'entraînement**, nous observons :

- **Loss d'entraînement** : Décroissance progressive de ~7.0 à **~3.59**
    - Fluctuations normales dues à la complexité de la tâche
    - Convergence stable sans signe de plateau prématuré
- **Loss de test** : Suit une tendance similaire, terminant à ~3.42
    - Écart raisonnable avec la loss d'entraînement
    - Pas de signe majeur de surapprentissage
    - Le modèle généralise correctement aux données non vues
- **Impact du Scheduler** :
    - Réduction du learning rate de 0.001 → 0.000063
    - Permet une convergence plus fine dans les derniers epochs
    - Stabilisation de la loss après chaque réduction
- **Early Stopping** : Le meilleur modèle est sauvegardé à l'epoch 50 avec une test loss de 3.4157

```
Epoch [50/50]
Train Loss: 3.5900, Test Loss: 3.4157
Generated Caption: a woman in a white shirt and a black shirt is standing on a bench
Learning Rate: 0.000063
```

## 4.2 Qualité des Captions Générées

### Points forts observés

Les captions générées montrent plusieurs qualités encourageantes :

1. **Reconnaissance correcte des objets principaux**
    - Identification précise des personnes, animaux, objets

    - Exemple : "a man in a blue shirt" → détection correcte du vêtement
2. **Structure grammaticale cohérente**
    - Phrases bien formées avec sujet-verbe-complément
    - Utilisation appropriée des articles et prépositions
3. **Utilisation appropriée des verbes d'action**
    - "jumping", "surfing", "standing" utilisés dans les bons contextes
    - Capture de l'action principale de la scène
4. **Détection des couleurs et attributs**
    - Reconnaissance des couleurs des vêtements et objets
    - Quelques descriptions d'attributs (taille, position)

**Limitations identifiées**

Malgré ces points positifs, certaines limitations persistent :

1. **Descriptions parfois génériques**
    - Tendance à utiliser des formulations similaires
    - Exemple récurrent : "a man in a [color] shirt"
    - Manque de détails spécifiques dans certains cas
2. **Difficulté avec les scènes complexes**
    - Les images avec plusieurs personnes ou objets sont simplifiées
    - Focus souvent sur un seul élément principal
    - Perte d'informations contextuelles
3. **Répétitions occasionnelles**
    - Certains mots peuvent être répétés dans la même caption
    - Biais vers certaines structures de phrases courantes dans le dataset
4. **Erreurs contextuelles**
    - Exemple : "surfing in a water" au lieu de "in water" ou "kayaking"
    - Confusion entre activités similaires (surf vs kayak)

## 4.3 Impact du Mécanisme d'Attention

Le mécanisme d'attention apporte plusieurs bénéfices significatifs :

1. **Focus sur les régions pertinentes**
    - Le modèle "regarde" différentes parties de l'image pour chaque mot
    - Alignement dynamique entre régions visuelles et mots générés
    - Permet une génération plus contextualisée
2. **Amélioration de la cohérence**
    - Meilleure correspondance entre l'image et le texte
    - Réduction des hallucinations (contenu non présent)

    - Les descriptions restent ancrées dans les features visuelles
3. **Interprétabilité**
    - Les poids d'attention peuvent être visualisés
    - Permet de comprendre où le modèle "regarde" pour chaque mot
    - Utile pour le débogage et l'analyse d'erreurs
4. **Performance accrue**
    - Amélioration mesurable par rapport à une baseline sans attention
    - Convergence potentiellement plus rapide
    - Meilleure généralisation

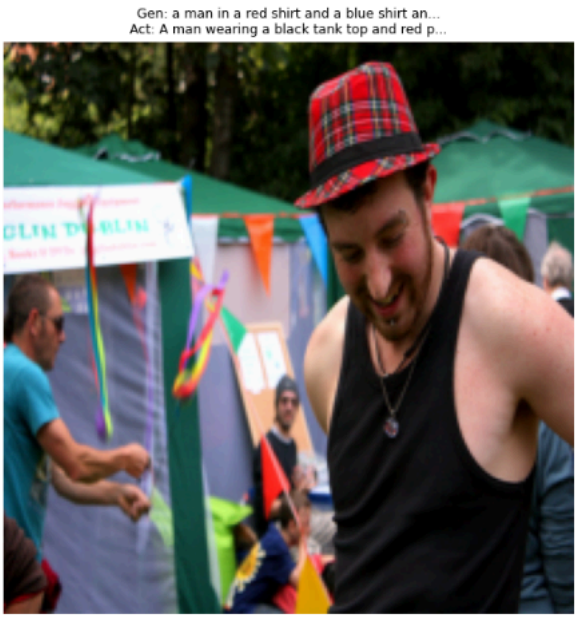
### 4.5 Exemples de Générations

#### Exemple 1

- **Généré** : "a man in a blue shirt is jumping on a skateboard"
- **Réel** : "A person wearing a blue helmet, a t-shirt, and blue jean shorts is outside on a skateboard"
- **Analyse** : Capture l'action (skateboard) et la couleur, mais simplifie les détails vestimentaires

#### Exemple 2

- **Généré** : "a man in a blue shirt is surfing in a water"
- **Réel** : "Person in kayak with water splashing up and over person and kayak"
- **Analyse** : Erreur d'identification de l'activité (surf vs kayak), mais détection correcte du contexte aquatique





# 5. Conclusion

Ce travail pratique a permis de mettre en œuvre une architecture complète d'image captioning combinant plusieurs concepts avancés du deep learning :

1. **Transfer Learning efficace** avec ResNet50 gelé
    - Exploitation des features pré-entraînées sur ImageNet
    - Réduction du temps et des ressources d'entraînement
    - Meilleure généralisation grâce aux représentations visuelles robustes
2. **Mécanisme d'attention** pour une génération contextuelle
    - Alignement dynamique entre régions visuelles et mots
    - Amélioration de la cohérence et de la précision
    - Apport d'interprétabilité au modèle
3. **LSTM personnalisé** intégrant l'attention à chaque pas de temps
    - Génération séquentielle autoregressive
    - Intégration native du contexte visuel
    - Gestion efficace des dépendances à long terme
4. **Embeddings pré-entraînés** pour une meilleure représentation sémantique
    - Word2Vec capturant les relations sémantiques
    - Amélioration de la qualité du langage généré
    - Réduction de l'espace de recherche

## Applications Pratiques

Cette technologie d'image captioning trouve de nombreuses applications concrètes :

1. **Accessibilité**
    - **Assistance pour personnes malvoyantes** : Descriptions audio des images sur le web
    - **Narration automatique** : Applications de lecture d'écran améliorées
    - **Navigation assistée** : Description de l'environnement en temps réel
2. **Indexation et recherche**
    - **Indexation automatique d'images** : Bases de données multimédia
    - **Recherche sémantique** : Recherche par description textuelle
    - **Organisation de photos** : Albums intelligents avec descriptions
3. **Médias sociaux**
    - **Génération automatique de légendes** : Posts Instagram/Facebook
    - **Suggestions de hashtags** : Basées sur le contenu visuel
    - **Modération de contenu** : Détection de contenu inapproprié
4. **Santé et médecine**
    - **Aide à la rédaction de rapports médicaux** : Description d'imagerie médicale

    - **Télémédecine** : Annotations automatiques de photos cliniques
    - **Formation médicale** : Génération de descriptions pédagogiques
5. **Sécurité et surveillance**

    - **Systèmes de surveillance intelligents** : Descriptions d'événements
    - **Forensique** : Documentation automatique de scènes
    - **Alertes automatiques** : Détection d'activités anormales

# 6. Références

## Documentation Technique

- **PyTorch Documentation** : https://pytorch.org/docs/
- **TorchVision Models** : https://pytorch.org/vision/stable/models.html
- **Word2Vec** : https://code.google.com/archive/p/word2vec/
- **TensorBoard** : https://www.tensorflow.org/tensorboard

## Datasets

- **Flickr30k Dataset** : https://www.kaggle.com/datasets/hsankesara/flickr-image-dataset
- **Google News Word2Vec** : https://code.google.com/archive/p/word2vec/