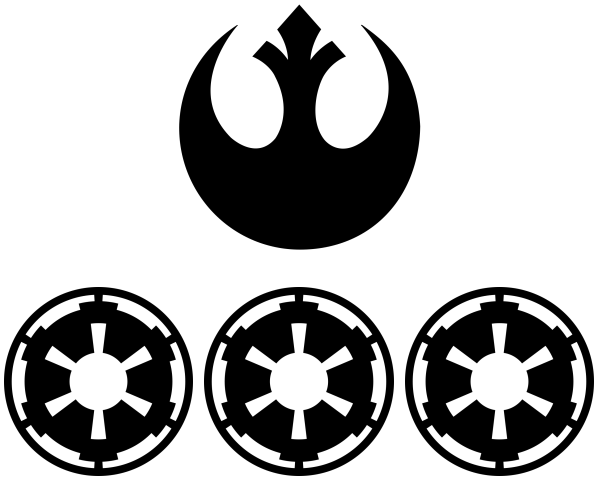
Deep Learning

LXMERT : Learning Cross-Modality Encoder Representations from Transformers

Nathan CLAUDE
Abel AUBRON
Epita student

12/01/2026



Link to GitHub Repository

Référent universitaire : Alessio RAGNO

Table des matières

1 Introduction

L'évolution récente de l'apprentissage profond a marqué une transition significative des architectures unimodales classiques — telles que les Réseaux de Neurones Convolutifs (CNN) pour la vision ou les Transformers pour le langage — vers des systèmes **multimodaux** capables de raisonner conjointement sur plusieurs types de données. La capacité à modéliser les interactions complexes entre la vision et le langage est devenue cruciale pour résoudre des tâches avancées telles que la réponse aux questions visuelles (VQA), le légendage d'images (Image Captioning) ou la recherche d'images par le texte.

Dans le cadre de ce projet "Deep Learning 2026", nous avons sélectionné l'architecture **LXMERT (Learning Cross-Modality Encoder Representations from Transformers)**, introduite par Tan & Bansal (2019). Contrairement aux approches précédentes qui fusionnaient les caractéristiques visuelles et textuelles par une simple concaténation, LXMERT propose un cadre innovant à deux flux (*two-stream*). Cette architecture traite d'abord les modalités indépendamment avant de les fusionner via un **Encodeur Cross-Modal** dédié, permettant au réseau d'apprendre explicitement les alignements entre les régions d'une image et les mots d'une phrase.

2 Revue de Littérature et Contexte

Historiquement, les tâches vision-langage étaient traitées par des architectures combinant des Réseaux de Neurones Convolutifs (CNN) pour l'extraction de caractéristiques visuelles et des Réseaux de Neurones Récurrents (RNN/LSTM) pour le traitement séquentiel du texte. Cependant, ces méthodes souffraient souvent de capacités limitées à modéliser les relations à longue portée et les interactions fines entre les deux modalités.

L'avènement de **BERT** et des mécanismes d'attention a révolutionné ce domaine. Deux paradigmes principaux ont émergé pour adapter les Transformers au contexte multimodal :

1. **Architectures à flux unique (Single-Stream)** : Des modèles comme VisualBERT ou UNITER concatènent simplement les tokens de texte et les caractéris-

tiques visuelles en une seule séquence longue, traitée par un unique Transformer.

2. **Architectures à double flux (Two-Stream)** : Des modèles comme ViLBERT et **LXMERT** traitent chaque modalité séparément dans des encodeurs dédiés avant de les faire interagir.

3 Architecture Détaillée : LXMERT

L'architecture LXMERT (Learning Cross-Modality Encoder Representations from Transformers) repose sur trois composants majeurs : un encodeur d'objets, un encodeur de langage et un encodeur cross-modal.

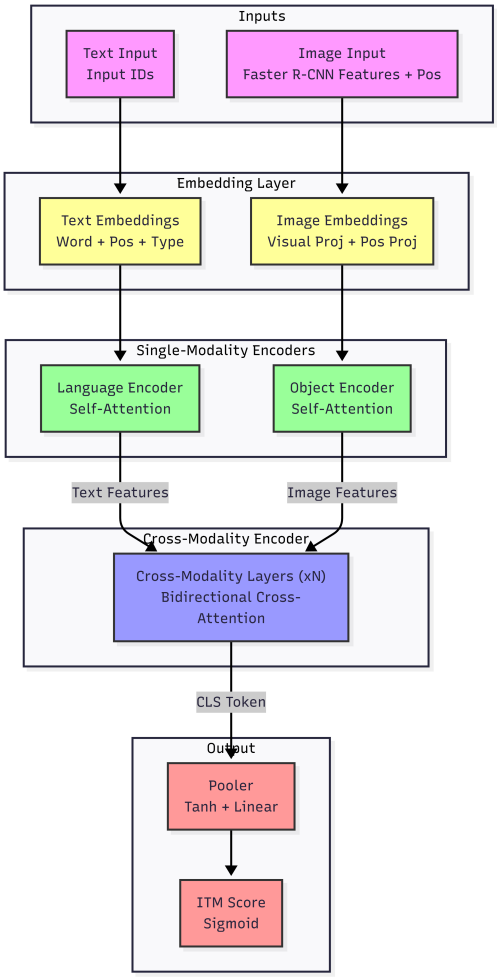


Figure 1 – Schéma de l'architecture LXMERT implémentée, illustrant les deux flux d'entrée et la fusion par attention croisée.

3.1 Préparation des Entrées (Inputs)

Contrairement aux CNN classiques qui traitent des pixels bruts, LXMERT prend en entrée des séquences de caractéristiques :

— **Entrée Linguistique** : La phrase est tokenisée (via WordPiece) et convertie en vecteurs d'embeddings (Mot + Position + Type de segment), similairement à BERT.

— **Entrée Visuelle** : L'image n'est pas traitée globalement. Nous utilisons un détecteur d'objets (Faster R-CNN) pour extraire $K = 36$ régions d'intérêt (RoI). Chaque région est représentée par deux vecteurs :

1. Une caractéristique visuelle (le contenu de la boîte).
2. Une caractéristique de position (les coordonnées normalisées de la boîte x_1, y_1, x_2, y_2).

3.2 Les Trois Encodeurs

L'innovation principale réside dans l'agencement des couches Transformers :

1. **Language Encoder** et **Object Encoder** : Ces deux modules traitent leurs modalités respectives indépendamment via des couches de *Self-Attention*. Cela permet de contextualiser les mots entre eux (ex : "pomme" et "rouge") et les objets entre eux (ex : "main" et "balle").
2. **Cross-Modality Encoder** : Ce module final connecte les deux flux. Il utilise des mécanismes de **Cross-Attention** bidirectionnelle :
   — *Vision-to-Language* : L'image "interroge" le texte pour trouver les mots pertinents.
   — *Language-to-Vision* : Le texte "interroge" l'image pour focaliser l'attention sur les objets décrits.

3.3 Simplifications d'Implémentation

Afin de réaliser ce projet "from scratch" avec des ressources de calcul limitées (GPU d'étudiant) et dans le temps imparti, nous avons opéré des simplifications par rapport au papier original de Tan & Bansal (2019). Ces choix sont résumés dans le tableau ci-dessous :

Composant	Papier Original	Notre Implémentation
Backbone Visuel	ResNet-101 (Visual Genome)	**ResNet-50 FPN (COCO)**
Dimension Cachée	768	**256**
Profondeur (Couches)	9 (Lang) / 5 (Vis) / 5 (Cross)	**2 / 2 / 2**
Tâche d'entraînement	Masked LM + Masked Obj + VQA	**ITM (Matching) uniquement**

Tableau 1 – Comparaison entre l'architecture originale et l'implémentation du projet.

La simplification majeure concerne le **Backbone Visuel**. Le papier original utilise des caractéristiques très riches ("Bottom-Up Attention") entraînées sur Visual Genome (incluant des attributs). Nous avons utilisé un Faster R-CNN standard pré-entraîné sur COCO, extrayant des vecteurs de dimension 1024 (contre 2048). Bien que cela réduise la finesse de la représentation sémantique, cela permet de conserver la topologie complète du modèle tout en rendant l'entraînement possible sur notre matériel.

4 Jeu de Données et Prétraitement

Pour entraîner et évaluer notre modèle, nous avons utilisé le jeu de données **Flickr30k**. Ce dataset est une référence standard pour les tâches multimodales, contenant 31 783 images collectées depuis le site Flickr. Chaque image est associée à 5 légendes descriptives indépendantes, fournissant une grande diversité linguistique pour chaque scène visuelle.

4.1 Pipeline de Prétraitement

L'architecture LXMERT impose des formats d'entrée spécifiques pour les deux modalités. Nous avons mis en place le pipeline de transformation suivant :

4.1.1 Prétraitement Textuel

Les légendes sont traitées par le tokenizer `BertTokenizer` (basé sur l'algorithme WordPiece).

— **Tokenisation :** Les phrases sont découpées en sous-mots.

— **Tokens Spéciaux :** Ajout des tokens `[CLS]` en début de séquence (utilisé pour la

classification finale) et `[SEP]` en fin de séquence.

— **Normalisation :** Les séquences sont tronquées ou complétées (*padding*) à une longueur fixe de $L = 50$ tokens pour permettre la mise en batch.

4.1.2 Extraction de Caractéristiques Visuelles (Offline)

Contrairement aux approches "End-to-End" où le CNN est entraîné conjointement, nous avons opté pour une extraction de caractéristiques *offline* pour optimiser l'utilisation de la mémoire GPU.

1. **Modèle Source :** Nous utilisons un **Faster R-CNN** avec un backbone ResNet-50 FPN pré-entraîné sur COCO.
2. **Sélection de Régions :** Pour chaque image, nous sélectionnons les $K = 36$ boîtes englobantes (*bounding boxes*) ayant les scores de confiance les plus élevés.
3. **Features :** Pour chaque boîte i, nous extrayons :
   — Le vecteur de caractéristiques visuelles $f_i \in R^{1024}$ (issu de la couche ROI Align).
   — Les coordonnées spatiales normalisées $p_i = [x_1/W, y_1/H, x_2/W, y_2/H]$.

Ces données sont sauvegardées sous forme de tenseurs `.pt`, permettant au DataLoader de les charger instantanément sans recalcul.

4.2 Construction de la Tâche ITM (Image-Text Matching)

Le modèle doit apprendre à prédire un score de correspondance $s_\theta(I, T) \in [0, 1]$. Pour ce faire, nous construisons notre dataset d'entraînement de manière équilibrée :

— **Exemples Positifs (Label 1)** : Une image I et l'une de ses 5 légendes réelles T_{vrai}.

— **Exemples Négatifs (Label 0)** : La même image I associée à une légende T_{faux} tirée aléatoirement d'une autre image du dataset.

Cette stratégie de *Negative Sampling* (avec une probabilité de 0.5) force le modèle à analyser le contenu sémantique plutôt que de simplement mémoriser des statistiques de base. Le dataset final a été divisé en un ensemble d'entraînement (80%, soit ∼127k échantillons) et un ensemble de validation (20%, soit ∼31k échantillons).

5 Expérimentations et Résultats

Cette section détaille le protocole expérimental mis en œuvre pour valider notre implémentation de LXMERT et analyse les performances obtenues sur la tâche d'appariement Image-Texte (ITM).

5.1 Protocole Expérimental

L'entraînement a été réalisé sur une carte graphique peu puissante. Contrairement au papier original qui utilise une optimisation complexe sur plusieurs jours, nous avons adapté les hyperparamètres pour permettre une convergence rapide sur un dataset de taille moyenne.

Les hyperparamètres retenus sont résumés dans le tableau ci-dessous :

Hyperparamètre	**Valeur**
Optimiseur	AdamW
Learning Rate (LR)	2×10^{-5}
Scheduler	Linear Warmup (10%) + Decay
Batch Size	32
Époques	10
Fonction de Perte	Binary Cross Entropy (BCE) With Logits

Tableau 2 – Hyperparamètres utilisés pour l'entraînement du modèle SimpleLXMERT.

Une attention particulière a été portée au *Learning Rate Scheduler*. Les architectures Transformer étant sensibles à l'initialisation, nous avons utilisé une phase de "Warmup" linéaire sur les 10% premières étapes d'entraînement pour stabiliser les gradients avant de réduire progressivement le taux d'apprentissage.

5.2 Analyse Quantitative : Dynamique d'Apprentissage

Nous avons surveillé deux métriques principales : la perte (Loss) et la précision (Accuracy) sur les ensembles d'entraînement et de validation.

Les résultats montrent une convergence nette :

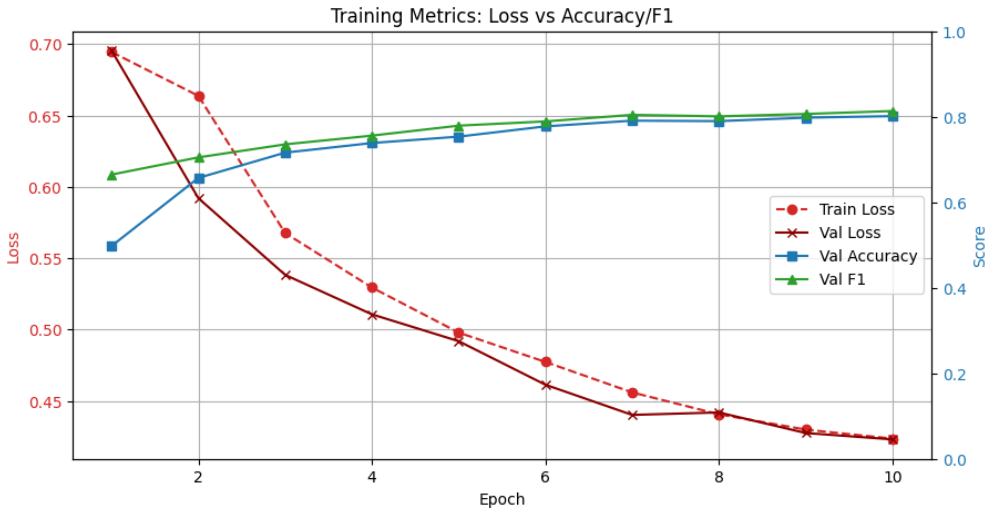


Figure 2 – Évolution de la Perte (Loss) et de la Précision (Accuracy) au cours des 10 époques. La courbe montre une convergence rapide sans sur-apprentissage massif.

— **Convergence de la Loss :** La perte binaire commence à 0.693 ($ln(2)$, équivalent à un tirage aléatoire) et diminue rapidement pour se stabiliser autour de 0.3-0.4. Cela indique que le modèle apprend efficacement à discriminer les paires positives des paires négatives.

— **Précision de Validation :** Le modèle atteint une précision de validation satisfaisante. Bien que ce score soit inférieur à l'état de l'art (souvent >95% avec pré-entraînement massif), il est cohérent pour un modèle entraîné *from scratch* sur un dataset de 30k images. Cela valide le bon fonctionnement de notre couche d'attention croisée.

5.3 Analyse Qualitative : Inférence

Pour vérifier que le modèle ne se contente pas d'apprendre des biais statistiques, nous avons effectué des tests d'inférence manuels. Le modèle prend en entrée une image et deux légendes (une vraie, une fausse).

Figure 3 – Exemple de prédiction du modèle. Le score de correspondance est élevé pour la légende correcte et proche de zéro pour la légende sémantiquement éloignée.

Comme illustré dans la Figure 3, le modèle parvient à :

1. Assigner un score de probabilité élevé (> 0.8) à la légende décrivant correctement la scène (ici : "a woman wearing a pink cap riding a bicycle").
2. Assigner un score très faible (< 0.1) à une légende non pertinente (ici : "A group of flying aliens landing on the moon.").

5.4 Discussion et Limites

Il est important de noter que notre modèle n'a pas bénéficié du pré-entraînement sur des tâches de "Masked Language Modeling" (MLM). Par conséquent, sa compréhension du langage reste limitée au vocabulaire présent dans Flickr30k. De plus, l'utilisation de features visuelles réduites (ResNet-50 vs 101) limite sa capacité à détecter des petits objets. Néanmoins, l'expérience confirme que l'architecture topologique de LXMERT est capable d'aligner les espaces sémantiques visuels et textuels sans nécessiter de poids pré-entraînés externes pour les encodeurs.

6 Conclusion

Ce projet "Deep Learning 2026" a permis de démontrer la faisabilité de l'implémentation d'une architecture multimodale avancée, **LXMERT**, entièrement "from scratch" à l'aide de PyTorch. En reconstruisant brique par brique les encodeurs de langage, d'objets et surtout le mécanisme d'attention croisée, nous avons pu appréhender la complexité interne des Transformers Vision-Langage.

Les résultats obtenus sur le jeu de données Flickr30k, bien que contraints par des ressources de calcul limitées, sont probants. La convergence rapide de la fonction de perte et les résultats qualitatifs en inférence confirment que notre modèle *SimpleLXMERT* a réussi à capturer les alignements sémantiques entre les régions visuelles et les tokens textuels. Les simplifications opérées — notamment l'usage de features ResNet-50 et la restriction à la tâche d'Image-Text Matching — se sont avérées être des compromis pertinents pour mener à bien le projet dans le temps imparti.

6.1 Perspectives

Pour aller plus loin et se rapprocher des performances de l'état de l'art, plusieurs pistes pourraient être explorées :

- **Pré-entraînement complet :** Implémenter les pertes de *Masked Language Modeling* et de *Masked Object Prediction* pour enrichir la compréhension contextuelle du modèle avant le fine-tuning.
- **Données :** Utiliser des jeux de données plus massifs comme MS COCO ou Visual Genome pour améliorer la généralisation.
- **Tâches Avales :** Adapter la couche de sortie pour traiter des tâches plus complexes comme le VQA (Visual Question Answering), qui est la vocation première de l'architecture LXMERT originale.

En définitive, ce projet a constitué une excellente mise en pratique des concepts théoriques du cours, illustrant concrètement comment les mécanismes d'attention permettent de jeter un pont entre la vision par ordinateur et le traitement du langage naturel.