Amélioration de la génération de légendes d'images avec RBM sur le module de captioning de BLIP (blip-image-captioning-base)

Fortuné HOUESSOU
Sous la supervision de M. Jérôme Lacaille

Mars 2025

Abstract

Le Image captioning, qui consiste à générer des descriptions textuelles à partir de contenus visuels, a connu des avancées majeures grâce aux modèles Vision-Language comme BLIP. Cependant, ces approches rencontrent encore des difficultés à produire des légendes à la fois précises et contextuellement cohérentes, notamment pour des images complexes ou ambiguës. Dans ce travail, nous proposons une amélioration innovante en intégrant une Machine de Boltzmann Restreinte (RBM) dans le processus de génération de légendes de BLIP. Les RBM, reconnues pour leur capacité à apprendre des représentations latentes profondes, permettent d'affiner la cohérence sémantique des légendes générées sans nécessiter un réentraînement massif sur de grandes bases de données. Nous explorons différentes stratégies d'intégration, où la RBM intervient pour réévaluer ou ajuster les sorties de BLIP, afin d'améliorer la pertinence et la richesse des descriptions. Des expériences approfondies menées sur des benchmarks standard montrent que cette approche hybride permet de générer des légendes plus naturelles et contextuellement adaptées, ouvrant ainsi de nouvelles perspectives pour l'amélioration des modèles Vision-Language.

Mots-clés : Captioning d'images, Modèles Vision-Langage (VLM), BLIP, Machine de Boltzmann Restreinte (RBM), Apprentissage profond.

Contents

1 Introduction

L'association entre la vision par ordinateur et le traitement automatique du langage naturel a révolutionné la manière dont les machines interprètent et décrivent le contenu visuel. Cette convergence a donné naissance à des modèles Vision-Language Models (VLMs), capables d'analyser une image et de générer une description textuelle cohérente et pertinente. L'un des modèles les plus performants dans ce domaine est le Bootstrapping Language-Image Pre-training (BLIP), qui exploite des techniques avancées d'apprentissage auto-supervisé pour améliorer la compréhension des relations entre les images et le texte.

Cependant, malgré son efficacité, BLIP souffre de certaines limitations, notamment une tendance à produire des descriptions génériques, parfois trop peu informatives ou manquant de diversité sémantique. Cette limitation est due en partie aux biais inhérents aux données d'entraînement et aux mécanismes d'optimisation utilisés. Pour remédier à ces faiblesses, il est crucial d'explorer des méthodes complémentaires qui puissent affiner les représentations latentes du modèle et améliorer la richesse du langage généré.

Dans cette optique, l'intégration d'un modèle probabiliste comme la Machine de Boltzmann Restreinte (RBM) représente une piste prometteuse. Les RBMs sont largement utilisées dans l'apprentissage non supervisé et la modélisation des distributions complexes. Elles offrent un moyen d'améliorer les représentations cachées en capturant des interactions non linéaires subtiles dans les données. En les combinant avec le module de captioning de BLIP, nous visons à enrichir la qualité des descriptions générées en affinant les représentations sémantiques sous-jacentes.

Ce travail explore donc l'intégration d'une RBM au sein du pipeline de génération de descriptions d'images de BLIP, en évaluant son impact sur la précision, la diversité et la pertinence linguistique des légendes produites. Cette approche pourrait ouvrir de nouvelles perspectives pour l'amélioration des modèles Vision-Language et leur application à des domaines exigeant une compréhension fine du contenu visuel, comme l'accessibilité numérique ou l'annotation automatique d'images.

2 Image - Captioning : État de l'Art

Le **captioning d'images**, ou annotation automatique d'images, est une tâche en vision par ordinateur et de langage consistant à générer une description textuelle cohérente et précise du contenu d'une image. Cette tâche est complexe car elle nécessite une compréhension approfondie des objets présents, de leurs actions, de leurs interactions et du contexte global de l'image.

2.1 Approches classiques du captioning d'images

Plusieurs approches ont été explorées parmi lesquelles :

2.1.1 Réseaux de Neurones Récurrents (RNN)

Les premières approches utilisaient des RNN, notamment des LSTM (Long Short-Term Memory), pour générer des descriptions séquentielles des images. Un pipeline typique impliquait l'utilisation d'un réseau de neurones convolutifs (CNN) pour extraire des caractéristiques visuelles de l'image, suivie d'un RNN pour générer la légende basée sur

ces caractéristiques. Par exemple, l'article *Language Models for Image Captioning: The Quirks and What Works* explore l'utilisation des RNN pour cette tâche. [**?**].

2.1.2 Transformers

Les modèles *Transformers*, introduits initialement pour le traitement du langage naturel, ont été adaptés au captioning d'images en raison de leur capacité à modéliser des dépendances à longue portée dans les données séquentielles. Ces modèles utilisent des mécanismes d'attention pour se concentrer sur différentes parties de l'image lors de la génération de chaque mot de la légende. L'article *A Review of Transformer-Based Approaches for Image Captioning* fournit une revue exhaustive des modèles de captioning d'images basés sur les Transformers[**?**].

2.1.3 Modèles Vision-Langage

Les modèles *Vision-Langage* intègrent des informations visuelles et textuelles pour améliorer la performance du captioning d'images. Ces modèles sont pré-entraînés sur de grandes quantités de données d'images et de textes associés, puis affinés pour des tâches spécifiques comme le captioning. Le modèle blip-image-captioning [8] qui fera l'objet de notre étude est un exemple concret des performances atteintes avec les VLMs.
Mais c'est quoi réellement un VLM ?

3 Vision-Language Models (VLM)

Les modèles de Vision Langage (VLM) sont des architectures multimodales qui combinent le traitement visuel et linguistique pour accomplir des tâches variées telles que la génération de légendes d'images, la recherche d'images par texte, ou l'association image-texte. Ces modèles ont gagné en popularité grâce à leur capacité à fusionner les informations textuelles et visuelles, permettant ainsi de résoudre des problèmes complexes en vision par ordinateur et en traitement du langage naturel. Parmi les modèles les plus connus dans ce domaine figurent CLIP [11] et BLIP [8].

BLIP, en particulier, adopte une approche de pré-entraînement utilisant des données web massives et combine un encodeur d'image avec un décodeur de texte pour générer des descriptions adaptées aux images. Cette approche a démontré son efficacité en génération de légendes et interprétation multimodale. De même, CLIP [11] se distingue par sa capacité à associer des images et des descriptions textuelles, facilitant ainsi des applications comme la recherche d'images par texte et l'analyse d'images à partir de requêtes textuelles.

Les VLMs sont également présents dans des travaux comme VisualBERT [9], qui fusionne les informations visuelles et textuelles dans un modèle transformeur pour des tâches de compréhension multimodale, telles que la réponse à des questions visuelles. Ce modèle a montré qu'il est possible d'aligner les représentations visuelles et textuelles dans un espace partagé pour des performances solides sur des tâches de vision et de langage.

Un autre modèle significatif dans ce domaine est ViLT [7], qui a introduit une approche simplifiée, ne recourant ni aux convolutions ni à une supervision régionale, permettant de traiter simultanément les images et les textes tout en réduisant la complexité du modèle. Cela a ouvert la voie à des applications plus légères tout en maintenant des performances compétitives.

Dans le cadre de la conduite autonome et d'autres applications de sécurité, des travaux comme VLM-RL [6] ont exploré l'intégration des VLMs avec l'apprentissage par renforcement pour améliorer la prise de décision basée sur des informations multimodales, notamment en gestion de scénarios visuels et textuels complexes.

Les capacités de raisonnement spatial sont également un domaine d'amélioration important pour les VLMs, comme le montre SpatialVLM [3], qui cherche à doter ces modèles de la capacité à effectuer des raisonnements géométriques et spatiaux, ce qui est essentiel pour des applications telles que la navigation autonome et la manipulation d'objets en 3D.

Enfin, des modèles comme VILA [5] et Xmodel-VLM [10] se concentrent sur des techniques de pré-entraînement et d'optimisation pour rendre les VLMs plus efficaces et adaptés aux environnements de production, en réduisant la taille des modèles tout en préservant leur capacité à traiter des informations visuelles et textuelles de manière précise.

Dans la suite de ce rapport sur l'État de l'Art des VLMs, de Blip et RBM, nous commençons par présenter de façon générale, le fonctionnement d'un VLM puis le role des mathématiques derrière ces modèles, et enfin nous traiterons les différents modèles qui feront l'objet de ce sujet à savoir BLIP (Bootstrapping Language-Image Pre-training) et RBM (Restricted Boltzmann machine).

Les **modèles de Vision et Langage** (VLMs) sont des systèmes qui essaient de comprendre à la fois des **images** et des **mots**. Imaginons qu'on montre une image de chat à un enfant et qu'on lui dise "C'est un chat". Le cerveau de l'enfant va faire le lien entre l'image du chat et le mots "chat". C'est exactement ce que font les VLMs, mais d'une manière beaucoup plus complexe et dans un ordinateur !

Comment ça marche concrètement ?

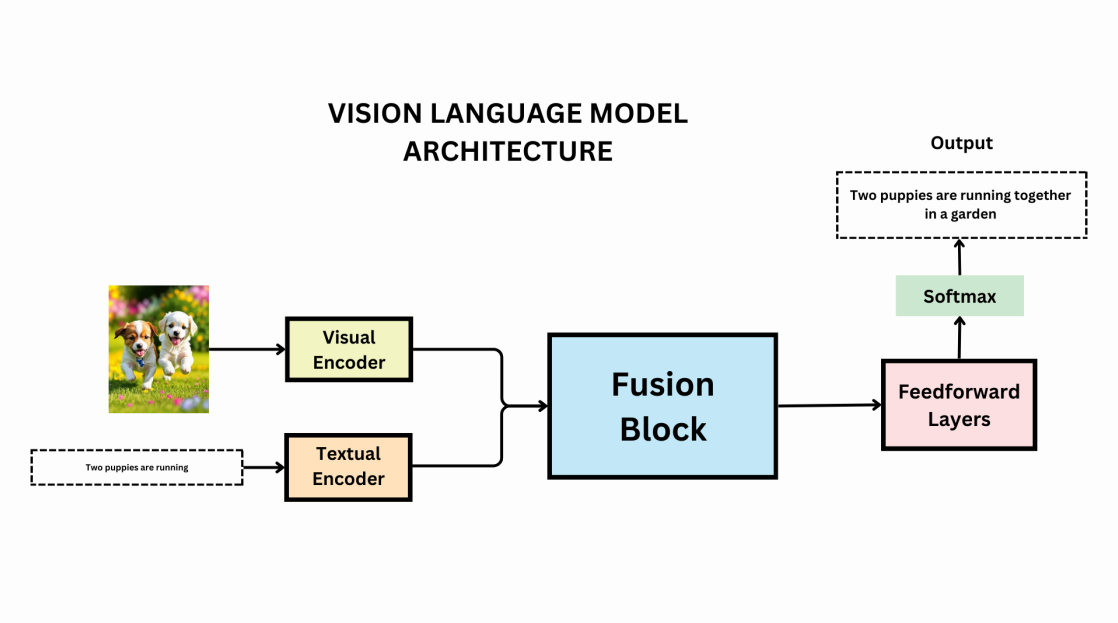

Figure 1: VLM

Le **VLM** combine la compréhension des **images** et des **mots**. Il analyse d'abord l'image pour en extraire des informations visuelles (formes, objets, etc.) via un réseau de neurones. Ensuite, les mots sont transformés en représentations que le modèle peut comprendre. Enfin, le VLM fusionne les informations des images et des mots pour accomplir des tâches comme l'association image-texte ou la génération de descriptions d'images. L'objectif est d'apprendre à lier efficacement ces deux modalités pour mieux comprendre et interagir avec le monde visuel et linguistique simultanément. C'est comme un Homme qui entends le mot "chat" et sait un peu à quelle forme s'attendre, parce que des chats il en a beaucoup vu, de différentes couleurs et dans différents horizons.

Un peu de formalisme :

L'entrainement du modèle BLIP s'éffectue en plusieurs étapes :

3.1 Prétraitement et Encodage

Soit x une entrée visuelle (image) et t une entrée textuelle (texte). Dans la plupart des VLMs, l'image x est d'abord passée à travers un encodeur d'image f_{image}, typiquement un réseau de neurones convolutifs (CNN) ou un modèle transformer ou vision encodeur adapté à l'image (par exemple, Vision Transformer, ViT). L'objectif est de projeter l'image dans un espace de représentation latente $\mathcal{Z}_{\text{image}} \in R^d$.

$$\mathcal{Z}_{\text{image}} = f_{\text{image}}(x)$$

De même, le texte t est encodé par un modèle de langage, tel que le Transformer, en une représentation vectorielle $\mathcal{Z}_{\text{texte}} \in R^d$. Le processus de traitement du texte suit un encodage similaire avec une fonction f_{texte}, qu'on ne connait pas à priori (BERT par exemple).

$$\mathcal{Z}_{\text{texte}} = f_{\text{texte}}(t)$$

Dans le processus d'embedding, certains modèles de VLM utilisent du : self-attention.

3.2 Fusion des Représentations : Alignement Visuel-Linguistique

Pour regrouper ces deux embeddings dans un même espace latent, il y a deux grandes stratégies :

A. Projection directe

On passe Z_{image} et Z_{texte} dans des Multi-Layer Perceptrons (MLP) [4] ou des couches linéaires afin de les projeter dans un **même espace latent**. Une fois cette projection effectuée, on calcule simplement la distance entre ces deux représentations, par exemple via une similarité:

$$\text{similarité}(Z_{\text{image}}, Z_{\text{texte}})$$

Les fonctions de similarité font l'objet d'une estimation de la distance entre deux embedding. C'est une façon de représenter par un nombre les différences ou rapprochements entre entités.

Fonctions de similarité

Basées sur la distance ou l'angle

- **Similarité cosinus (Cosine Similarity)** :

$$\text{sim}(x, y) = \frac{x \cdot y}{\|x\|\|y\|}$$

- **Distance euclidienne (Euclidean Distance)** :

$$d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

  Souvent transformée en :

$$\text{sim}(x, y) = \frac{1}{1 + d(x, y)}$$

Basées sur des probabilités

- **Divergence de Kullback-Leibler (KL Divergence)** :
  Pour deux distributions de probabilités P et Q :

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log \left(\frac{P(i)}{Q(i)} \right)$$

  Pas symétrique et non bornée.

- **Divergence de Jensen-Shannon (JS Divergence)** :
  Variante symétrique et plus stable du KL :

$$D_{JS}(P, Q) = \frac{1}{2} D_{KL}(P \parallel M) + \frac{1}{2} D_{KL}(Q \parallel M)$$

  où

$$M = \frac{1}{2}(P + Q)$$

Basées sur des ensembles

- **Similarité de Jaccard (Jaccard Similarity)** :
  Pour des ensembles A et B :

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- **Coefficient de recouvrement (Overlap Coefficient)** :

$$\text{Overlap}(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)}$$

C'est précisément l'approche adoptée par le modèle CLIP.

B. Cross-Attention

Dans certains modèles comme BLIP, avant la projection des données bimodales (image/texte) dans l'espace latent,un mecanisme d'attention croisée (cross-attention) est utilisé dans les architectures de type transformer. Ce mécanisme permet de calculer une attention mutuelle entre les représentations des images et du texte afin d'extraire des correspondances pertinentes entre les modalités. Les articles [2] et [1] introduisent et expliquent dans les détails le mécanisme d'attention. Le calcul de l'attention croisée peut être formalisé comme suit :

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

où Q (Query) est la requête (environnement textuel ou visuel), K (Key) est la clé (respectivement l'information visuelle ou textuelle), et V (Value) est la valeur : [14] et [13]. Ce mécanisme est appliqué pour apprendre les relations entre $\mathcal{Z}_{\text{image}}$ et $\mathcal{Z}_{\text{texte}}$, en ajustant les poids des connexions neuronales selon l'importance de chaque élément textuel ou visuel pour le modèle.

Optimisation et Apprentissage

L'entraînement des VLMs repose sur la minimisation d'une fonction de perte $\mathcal{L}$ qui permet au modèle d'apprendre à associer de manière optimale les informations visuelles et textuelles. Par exemple, une fonction de perte typique dans le cas d'une tâche de classification d'images par texte est la suivante :

$$\mathcal{L} = \sum_i \left[\log P(y_i|\mathcal{Z}_{\text{image}}) + \log P(y_i|\mathcal{Z}_{\text{texte}})\right]$$

où y_i est l'étiquette associée à l'entrée x_i ou t_i. Cette formulation implique une double minimisation des erreurs sur les prédictions visuelles et textuelles.

Un autre type de perte, utilisé dans les modèles de type *contrastive learning* (comme CLIP [11]), repose sur l'idée de maximiser la similarité entre les représentations textuelles et visuelles d'une même instance tout en minimisant la similarité entre les paires d'images et de textes différentes. La fonction de perte contraste peut être formulée ainsi :

$$\mathcal{L}_{\text{contrastive}} = -\sum_i \log \frac{\exp(\text{sim}(\mathcal{Z}^i_{\text{image}}, \mathcal{Z}^i_{\text{texte}}))}{\sum_j \exp(\text{sim}(\mathcal{Z}^i_{\text{image}}, \mathcal{Z}^j_{\text{texte}}))}$$

où $\text{sim}(a, b)$ représente la similarité entre les vecteurs a et b, souvent mesurée par un produit scalaire ou un cosinus.

Architecture générale d'un VLM (Vision-Language Model)

1 Embedding

- Image → Patchs (ViT) ou CNN → Embeddings visuels.
- Texte → Tokenisation → Embeddings textuels.

2 Self-Attention(selon le modèle)

- Appliqué séparément sur les embeddings :
- Image → Self-attention entre patchs.
- Texte → Self-attention entre tokens.

3 Cross-Attention (si fusion, cas de BLIP)

- Texte attend sur l'image ou inversement.
- Q = texte, K, V = image (ou l'inverse selon le modèle).
- Permet d'associer finement les informations. (On dit que le texte et l'image se regardent)

4 Projection

- Alignement dans un même espace vectoriel via couches linéaires.
- Image → projection.
- Texte → projection.

5 Similarité

- Calcul par produit scalaire ou cosin similarity entre vecteurs projetés.

6 Contrastive Loss

- But :
  - Maximiser la similarité des bonnes paires (image, texte liés).
  - Minimiser celle des paires incorrectes.

"""

4 BLIP (Bootstrapping Language-Image Pre-training)

BLIP (*Bootstrapping Language-Image Pre-training*) [8] est un framework conçu pour améliorer les performances sur les tâches vision-langage. Il se distingue par sa capacité à exploiter à la fois des **données bruitées web-scale** et des annotations de qualité, grâce à une architecture flexible et un apprentissage par bootstrapping.

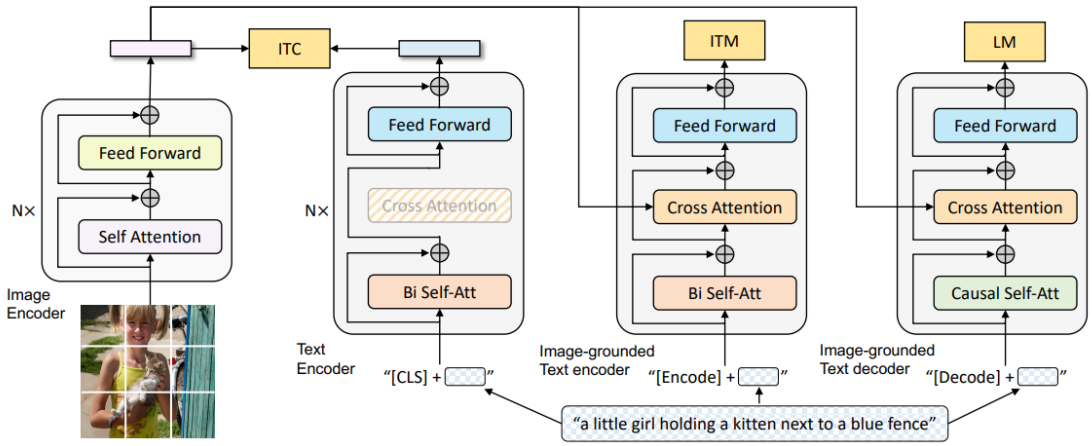

Figure 2: BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation

4.0.1 Fonctionnement général

BLIP repose principalement sur trois tâches de pré-entraînement complémentaires :

- **Image-Text Contrastive Learning (ITC)** : maximise la similarité entre les représentations d'une image et de sa légende correspondante, et minimise celles des paires non correspondantes.
  L'ITC est une fonction de perte utilisée pour aligner les représentations des images et des textes dans un espace commun. Elle est inspirée de la InfoNCE loss et fonctionne comme une perte contrastive entre les paires image-texte positives et négatives.

$$\mathcal{L}_{ITC} = -\frac{1}{N}\sum_{i=1}^{N}\left(\log\frac{e^{\text{sim}(z_i^I, z_i^T)/\tau}}{\sum_{j=1}^{N} e^{\text{sim}(z_i^I, z_j^T)/\tau}}\right) \tag{1}$$

  où z_i^I et z_i^T sont les embeddings d'image et de texte, sim est la similarité cosinus, τ est la température (temperature scaling) et N la taille du lot des données traitées.

- **Image-Text Matching (ITM)** : discrimine si une paire image-texte est correcte ou incorrecte via une classification binaire, souvent couplée à une fonction de perte d'entropie croisée classique.

$$\mathcal{L}_{ITM} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log P(y_i|I_i, T_i) + (1-y_i)\log(1 - P(y_i|I_i, T_i))\right]$$

- **Image-Conditioned Language Modeling (LM)** : génère du texte conditionnellement à une image.

$$\mathcal{L}_{LM} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i}\log P(w_t^i|w_{<t}^i, I_i)$$

où :

- N est le nombre d'exemples dans le batch.
- T_i est la longueur de la séquence de texte pour l'exemple i. (nombre de token)
- w_t^i est le mot à la position t dans la séquence i.
- $w_{<t}^i$ représente tous les mots précédents dans la séquence.
- I_i est l'image associée.
- $P(w_t^i \mid w_{<t}^i, I_i)$ est la probabilité prédite du mot w_t^i étant donné l'image et les mots précédents.

4.0.2 Architecture globale

BLIP s'appuie sur une combinaison de :

- **Encoders visuels** (souvent des Vision Transformers - ViT)
- **Encoders textuels** (type BERT)
- **Décoders textuels** (transformer auto-régressif)

Pendant l'entrainement, les modèles apprennent dans un espace latent partagé où les embeddings d'images et de textes sont projetés, permettant une compatibilité sémantique entre les deux modalités.

Dans ce projet, nous nous interessons plus précisément au modèle BLIP-Captionner du framework BLIP, qui en sortie, génère une description textuelle en réponse à une image reçue en entrée.

5 Blip-Image-Captioning-Base

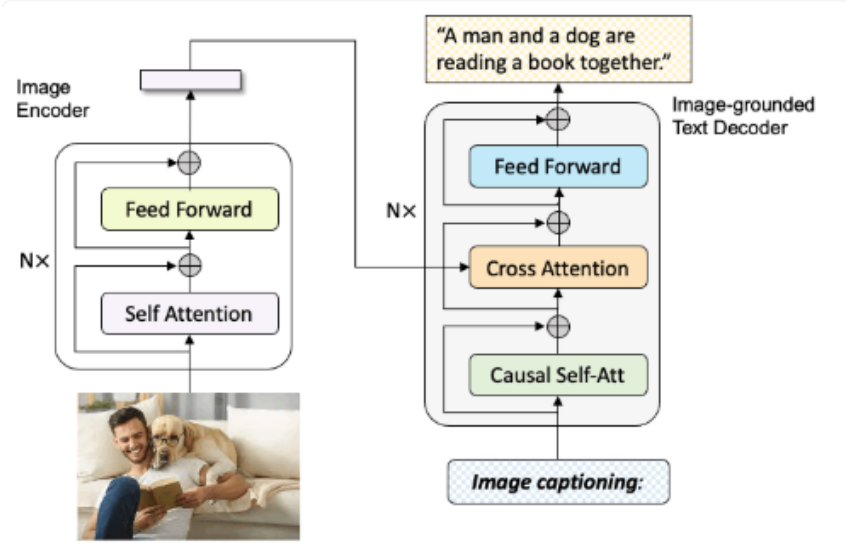

Figure 3: Blip-image-captioning-base

Pour la génération de légende d'image (image captioning) le modele blip-image-captioning-base utilse :

- Un encodeur d'image (ViT) composé de couches d'auto attention (SA) et d'un reseau Feed Forward (FF) le tout entrainé sur des bases de données annotées par des humains comme Coco et des données bruitées du web et néttoyées par bootstrapping via la méthode CapFilt. CapFilt est un ensemble formé d'un Captioner(Image-grounded Text Decoder)[8] et d'un Flilter(Image-grounded Text Encoder) qui sont des modules du framework BLIP et qui servent à reannoter (génération de légende) les images bruitées obtenues sur le web.

- Le décodeur de Blip-image-captioning-base est composé d'un réseau FF, pré-entrainé avec de l'auto attention bidirectionnelle sur coco dataset. La perte utilisée est l'ITC (confère figure 2). Il n'y a pas d'attention croisée (le module "Cross Attention" est gelé dans un premier temps).Dans un second temps, le meme modèle pré-entrainé est repris et cette fois ci avec l'attention croisée(image-grounded Text encodeur sur la figure 2) et reentrainé sur Coco dataset pour une tache de classification binaire [8]. La perte utilisée est l'ITM. Enfin le décodeur de blip-image-captioning est obtenu en remplaçant les couches d'auto attention bidirectionnelle par des couches d'auto-attention causale comme sur la figure 2 (image-grounded Text Decoder). La perte utilisée lors de l'entrainement est la LM.

Notre objectif est d'augmenter les performances de blip-image-captioning pour affiner la génération de légende d'image par insertion d'une RBM (Machine de Blotzmann Restreinte) entre l'encodeur et le décodeur du modele.

6 La Machine de Boltzmann Restreinte (RBM)

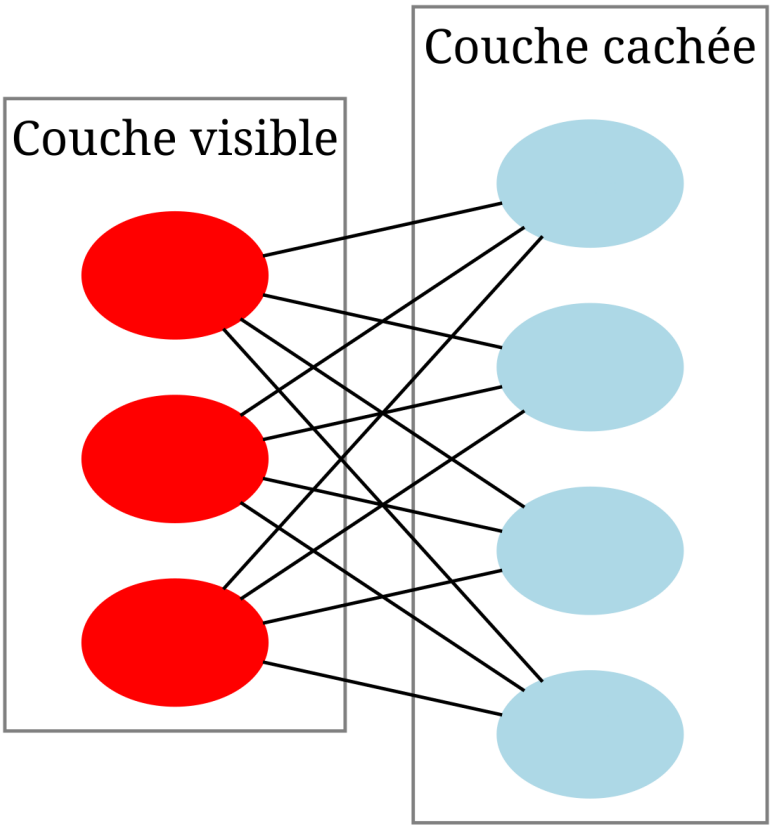

Figure 4: Restricted Boltzmann machine

La Machine de Boltzmann Restreinte (RBM), introduite par [12] et popularisée pour l'apprentissage non supervisé, est un modèle probabiliste génératif appartenant à la famille des modèles énergétiques. Apparues dans les années 1980, les Machines de Boltzmann Restreintes (RBM) ont bénéficié d'une meilleure attention avec l'essor des capacités de calcul et leur intégration dans des structures d'apprentissage profondes, en particulier les Deep Belief Networks (DBN). En superposant plusieurs RBM, il devient possible d'extraire des représentations successives des données, passant de caractéristiques élémentaires à des concepts plus abstraits. Par ailleurs, lorsqu'une RBM est exploitée pour transformer des entrées en représentations latentes, elle peut être assimilée à un réseau de neurones à propagation avant, optimisant ainsi son intégration dans des modèles supervisés pour améliorer l'efficacité de l'apprentissage.

6.1 Pourquoi la machine de Boltzmann restreinte ?

L'utilisation d'un **Restricted Boltzmann Machine (RBM)**pour améliorer l'**image captioning** est un choix intuitif et judicieux, notamment en raison de sa capacité à apprendre des représentations riches et compactes des données visuelles de manière **non supervisée**. En effet, un RBM permet de modéliser efficacement une distribution de probabilité inconnue en découvrant des caractéristiques latentes pertinentes dans les

images. Grâce à sa structure bipartite composée de **variables visibles** (les pixels ou des features extraites d'un réseau de neurones) et de **variables latentes** (invisibles), il est capable de capturer des **dépendances complexes** entre les pixels d'une image et d'encoder des relations de haut niveau, facilitant ainsi la génération de descriptions précises et cohérentes.

Dans un cadre d'apprentissage probabiliste, le RBM optimise ses **paramètres** θ via des méthodes comme **l'estimation du maximum de vraisemblance**, ce qui lui permet d'encoder efficacement la structure sous-jacente des images. Cette approche est particulièrement bénéfique pour l'image captioning, où il est crucial de comprendre non seulement les objets présents dans une image, mais aussi leurs relations contextuelles et sémantiques. En apprenant des représentations de manière non supervisée, le RBM permet de généraliser sur de nouvelles images et d'améliorer la robustesse du modèle de génération de descriptions textuelles.

Enfin, en intégrant un RBM en amont d'un modèle de génération de texte, on bénéficie d'une représentation d'entrée plus expressive et informative, ce qui améliore la qualité des légendes générées. Cette synergie entre **modélisation probabiliste** et **apprentissage profond** fait du RBM un choix judicieux et intuitif pour l'amélioration des systèmes d'image captioning.

6.2 Structure d'une RBM

Une RBM est un réseau biparti composé de :

- **Une couche visible** $\mathbf{v} = (v_1, v_2, \ldots, v_m)$ représentant les données observables.
- **Une couche cachée** $\mathbf{h} = (h_1, h_2, \ldots, h_n)$ permettant de capturer des dépendances complexes (cachées).

Contrairement aux Machines de Boltzmann standards, il n'y a **pas de connexions intra-couche** :

- Aucune connexion entre les neurones visibles.
- Aucune connexion entre les neurones cachés.

6.3 Fonction énergétique

Le cœur de la RBM repose sur une fonction d'énergie définie par :

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^\top \mathbf{W}\mathbf{h} - \mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h}$$

où :

- $\mathbf{W}$ est la matrice des poids entre les couches visible et cachée.
- $\mathbf{b}$ et $\mathbf{c}$ sont les biais des couches visible et cachée respectivement.
- $\mathbf{v}$ et $\mathbf{h}$ sont les états des neurones sur les couches visible et cachée respectivement.

6.4 La RBM comme modèle probabiliste

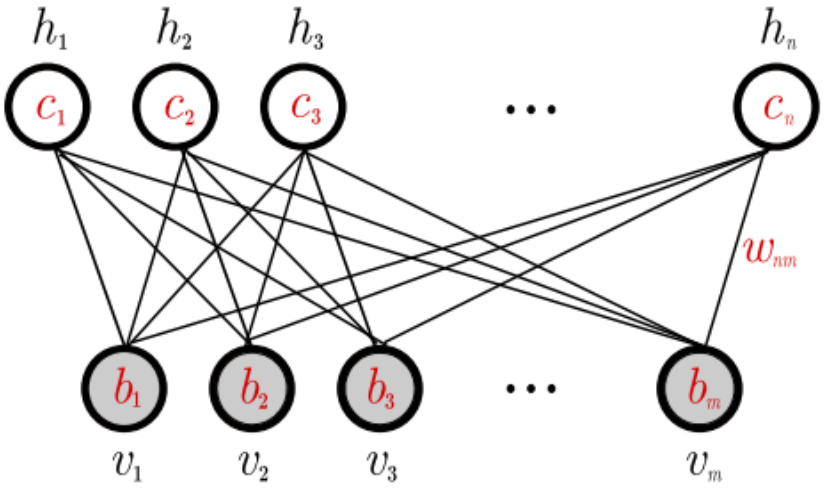

Figure 5: Restricted Boltzmann machine

Une Machine de Boltzmann Restreinte (RBM) est un modèle probabiliste basé sur un champ de Markov aléatoire structuré sous forme d'un graphe biparti non orienté. Elle est constituée de m unités visibles $V = (V_1, ..., V_m)$, qui représentent les données observables, et de n unités cachées $H = (H_1, ..., H_n)$, chargées de capturer les interactions entre ces données.

Dans le cadre des RBM binaires, qui sont l'objet de cette étude, les variables aléatoires (V, H) prennent des valeurs dans $\{0, 1\}^{m+n}$. La distribution conjointe associée à ce modèle suit la loi de Gibbs :

$$p(v, h) = \frac{1}{Z} e^{-E(v,h)}$$

avec l'énergie définie par :

$$E(v, h) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i.$$

Ici, pour tout $i \in \{1, ..., n\}$ et $j \in \{1, ..., m\}$, w_{ij} désigne un coefficient réel associé à la connexion entre l'unité cachée H_i et l'unité visible V_j. Les paramètres b_j et c_i correspondent respectivement aux biais appliqués aux unités visibles et cachées.

L'entraînement d'une **Machine de Boltzmann Restreinte (RBM)** repose principalement sur des méthodes d'optimisation adaptées aux modèles probabilistes à base d'énergie. L'algorithme d'apprentissage le plus couramment utilisé est **Contrastive Divergence (CD)**, introduit par Geoffrey Hinton. D'autres algorithmes comme : Parallel Tempering et Persistent Contrastive Divergence (PCD) sont aussi utilisés.

L'apprentissage des paramètres $(\mathbf{W}, \mathbf{b}, \mathbf{c})$ repose sur une estimation approchée du gradient du logarithme de la vraisemblance :

$$\Delta W = \epsilon \cdot (\langle v h^T \rangle_{\text{données}} - \langle v h^T \rangle_{\text{modèle}}) \tag{2}$$

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model})$$

où :

- ϵ est le taux d'apprentissage.
- $\langle \cdot \rangle_{data}$ est l'espérance sous la distribution empirique des données.
- $\langle \cdot \rangle_{model}$ est l'espérance sous la distribution modélisée.

Le processus implique des étapes de **Gibbs Sampling**, alternant entre la mise à jour de **h** et **v** :

$$P(h_j = 1|\mathbf{v}) = \sigma\left(\sum_i w_{ij} v_i + c_j\right)$$

On peut reécrire :

$$P(h_j = 1|\mathbf{v}) = \sigma\left(\sum_i w_{ij} v_i + \frac{c_j}{w_0} w_0\right) = \sigma\left(\sum_i w_{ij} v_i + c'_j w_0\right)$$

$$P(v_i = 1|\mathbf{h}) = \sigma\left(\sum_j w_{ij} h_j + b_i\right)$$

où $\sigma(x)$ est la fonction sigmoïde.

6.5 Formalisme

vraissemblance du modèle Posons

$$\mathcal{L}(\theta|v) = p(v|\theta) = \frac{1}{Z} \sum_h e^{-E(v,h)}$$

par indépendance des variables v sachant h on a :

$$\prod_{v \in V} \mathcal{L}(\theta|v)$$

La log vraissemblance :

$$\ln \mathcal{L}(\theta|v) = \ln p(v|\theta) = \ln \frac{1}{Z} \sum_h e^{-E(v,h)} = \ln \sum_h e^{-E(v,h)} - \ln \sum_{v,h} e^{-E(v,h)}$$

Gradient de la log vraissemblance :

$$\frac{\partial \ln \mathcal{L}(\theta|v)}{\partial \theta} = \frac{\partial}{\partial \theta}\left(\ln \sum_h e^{-E(v,h)}\right) - \frac{\partial}{\partial \theta}\left(\ln \sum_{v,h} e^{-E(v,h)}\right)$$

$$= -\frac{1}{\sum_h e^{-E(v,h)}} \sum_h e^{-E(v,h)} \frac{\partial E(v,h)}{\partial \theta} + \frac{1}{\sum_{v,h} e^{-E(v,h)}} \sum_{v,h} e^{-E(v,h)} \frac{\partial E(v,h)}{\partial \theta}$$

et en utilisant l'égalité :

$$p(h|v) = \frac{p(v,h)}{p(v)} = \frac{\frac{1}{Z}e^{-E(v,h)}}{\frac{1}{Z}\sum_h e^{-E(v,h)}} = \frac{e^{-E(v,h)}}{\sum_h e^{-E(v,h)}}$$

on a :

$$\frac{\partial \ln \mathcal{L}(\theta|v)}{\partial \theta} = -\sum_h p(h|v)\frac{\partial E(v,h)}{\partial \theta} + \sum_{v,h} p(v,h)\frac{\partial E(v,h)}{\partial \theta}(1)$$

$$\frac{1}{N}\sum_{v\in S}\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|v)}{\partial \theta} = \frac{1}{N}\sum_{v\in S}\left[-E_{p(h|v)}\left[\frac{\partial E(v,h)}{\partial \theta}\right] + E_{p(h,v)}\left[\frac{\partial E(v,h)}{\partial \theta}\right]\right](2)$$

$$\frac{1}{N}\sum_{v\in S}\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|v)}{\partial \theta} = \frac{1}{N}\sum_{v\in S}\left[-E_{données}\left[\frac{\partial E(v,h)}{\partial \theta}\right] + E_{modele}\left[\frac{\partial E(v,h)}{\partial \theta}\right]\right](3)$$

On remplace le modèle $\theta = (W, b, c)$

$$\sum_h p(h \mid v)\frac{\partial E(v,h)}{\partial w_{ij}} = \sum_h p(h \mid v)h_i v_j$$

$$= \sum_h \prod_{k=1}^{n} p(h_k \mid v)h_i v_j = \sum_{h_i}\sum_{h_{-i}} p(h_i \mid v)p(h_{-i} \mid v)h_i v_j$$

$$= \sum_{h_i} p(h_i \mid v)h_i v_j \sum_{h_{-i}} \underbrace{p(h_{-i} \mid v)}_{=1} = p(H_i = 1 \mid v)v_j = \sigma\left(\sum_{j=1}^{m} w_{ij}v_j + c_i\right)v_j(4)$$

Pour la moyenne de cette dérivée sur un ensemble d'apprentissage $S = \{\mathbf{v}_1, \ldots, \mathbf{v}_N)$ les notations suivantes sont souvent utilisées :

$$\frac{1}{N}\sum_{v\in S}\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \mid v)}{\partial w_{ij}} = \frac{1}{N}\sum_{v\in S}\left[-E_{p(h|v)}\left[\frac{\partial E(v,h)}{\partial w_{ij}}\right] + E_{p(h,v)}\left[\frac{\partial E(v,h)}{\partial w_{ij}}\right]\right]$$

$$= \frac{1}{N}\sum_{v\in S}\left[E_{p(h|v)}[v_i h_j] - E_{p(h,v)}[v_i h_j]\right](5)$$

Par analogie on a;

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \mid v)}{\partial b_i} = v_j - \sum_v p(v)v_j(6)$$

et

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \mid v)}{\partial c_i} = p(H_i = 1 \mid v) - \sum_v p(v)p(H_i = 1 \mid v)(7)$$

Approximation de la log vraissemblance:

L'idée ici est d'éviter un calcul trop complexe qui nécessiterait de parcourir toutes les valeurs possibles des variables visibles (complexité exponentielle). Ce problème apparaît notamment lorsqu'on calcule le gradient du log-vraisemblance, en particulier pour certains termes spécifiques des équations (5), (6) et (7). Les embeddings récupérer en sortie de l'encodeur de Blip sont de tailles $(577)(768) = 443136$, donc il faudrait évaluer 2^{443136} possibilités. Celà est juste impossible car meme le supercalculateur le plus puissant de notre ère ne finirait ce calcul avant des milliers d'années.

Solution possible

Plutôt que d'effectuer ces sommes directement, on peut estimer ces valeurs en prenant des échantillons générés à partir du modèle lui-même. Pour obtenir ces échantillons, on utilise une méthode appelée échantillonnage de Gibbs, qui repose sur une chaîne de Markov. Cette chaîne doit tourner assez longtemps pour atteindre un état stable.

Cependant, même avec cette méthode, le coût de calcul reste trop élevé pour un apprentissage efficace des RBM. C'est pourquoi des approximations supplémentaires sont souvent utilisées, comme la CD-k (k-Contrastive Divergence).
La divergence contrastive (CD) simplifie l'entraînement des RBM en limitant le nombre d'itérations de l'échantillonnage. La chaîne est initialisée avec un exemple d'entraînement $v(0)$ et génère un échantillon $v(k)$ après k étapes. À chaque étape, on commence par échantillonner $h(t)$ à partir de $p(h|v(t))$, puis $v(t+1)$ à partir de $p(v|h(t))$. Cette approximation permet d'accélérer le calcul du gradient et l'optimisation du modèle.

Algorithm 1 k-step contrastive divergence

Input: RBM $(V_1, \ldots, V_m, H_1, \ldots, H_n)$, training batch S
Output: gradient approximation Δw_{ij}, Δb_j, and Δc_i for $i = 1, \ldots, n$, $j = 1, \ldots, m$

Function `k-step contrastive divergence`$(V_1, \ldots, V_m, H_1, \ldots, H_n, S)$`:`
- **for** $i = 1, \ldots, n$, $j = 1, \ldots, m$ **do**
  - $\Delta w_{ij} = 0$, $\Delta b_j = 0$, $\Delta c_i = 0$
- **end**
- **for** *each* $v \in S$ **do**
  - $v^{(0)} \leftarrow v$
  - **for** $t = 0, \ldots, k-1$ **do**
    - **for** $i = 1, \ldots, n$ **do**
      - sample $h_i^t \sim p(h_i|v^{(t)})$
    - **end**
    - **for** $j = 1, \ldots, m$ **do**
      - sample $v_j^{(t+1)} \sim p(v_j|h^{(t)})$
    - **end**
  - **end**
  - **for** $i = 1, \ldots, n$, $j = 1, \ldots, m$ **do**
    - $\Delta w_{ij} \leftarrow \Delta w_{ij} + p(H_i = 1|v^{(0)}) \cdot v_j - p(H_i = 1|v^{(k)}) \cdot v_j$ $\Delta b_j \leftarrow \Delta b_j + v_j^{(0)} - v_j^{(k)}$
    - $\Delta c_i \leftarrow \Delta c_i + p(H_i = 1|v^{(0)}) - p(H_i = 1|v^{(k)})$
  - **end**
- **end**

Nous avons introduire la RBM binaire car c'est de là que part toutes les variantes existantes de la Machine de Boltzmann Restreinte. Les embeddings obtenues en sortie de l'encodeur d'image du modèle Blip-Image-Captioning étant des valeurs continues (pas que des 0 et 1), nous allons travailler avec la RBM gaussienne.

7 Qu'est-ce qu'une RBM Gaussienne-Binaire?

C'est une **extension** de la RBM binaire qui permet de **traiter des données continues** (comme des embeddings, des signaux audio, ou des images en niveaux de gris).

Différence clé :

- Dans une RBM **binaire**, les neurones visibles prennent **0 ou 1**.
- Dans une RBM **gaussienne**, les neurones visibles prennent **des valeurs continues**.

Pourquoi c'est utile ? Parce que dans ce sujet les embeddings sont **des vecteurs de nombres réels**, donc une RBM gaussienne serait plus adaptée qu'une RBM binaire.

8 Fonctionnement d'une RBM-GB

Elle garde la **même structure** qu'une RBM classique, mais avec une **modification sur les états des neurones visibles**.

8.1 Activation des neurones cachés (identique à la RBM Binaire-Binaire)

Chaque neurone caché h_j reçoit une somme pondérée des entrées v_i et applique une **sigmoïde** :

$$P(h_j = 1|v) = \sigma\left(\sum_i w_{ij} v_i + b_j\right)$$

Donc les **neurones cachés restent binaires (0 ou 1)**, comme dans une RBM classique.

8.2 Activation des neurones visibles (modifié pour les valeurs continues)

Dans une RBM binaire, on n'applique pas une sigmoïde aux **neurones visibles**. Mais ici, on considère qu'ils suivent une **distribution gaussienne** :

$$v_i \sim \mathcal{N}\left(c_i + \sum_j w_{ij} h_j, \sigma^2\right)$$

(σ est un hyperparametre.)
Chaque v_i est un **nombre réel** tiré d'une **distribution normale (gaussienne)** centrée autour de $\sum_j w_{ij} h_j + c_i$, avec une variance σ^2.

Ce que ça signifie :

- Au lieu d'être **0 ou 1**, les neurones visibles prennent **des valeurs réelles**.
- On peut interpréter ça comme une **reconstruction bruitée, décompressée ou raffinée** des embeddings.

9 Entraînement d'une RBM gaussienne

L'apprentissage suit la même logique qu'une RBM binaire :

1. On passe une **donnée réelle** dans la RBM.
2. On **calcule les activations cachées** avec la sigmoïde.
3. On **reconstruit les neurones visibles** avec une distribution gaussienne.
4. On ajuste les poids avec le **Contraste de Divergence (CD-k)**.

Petite différence mais importante : Comme les neurones visibles sont continus, la mise à jour des poids prend en compte la variance σ^2, ce qui change un peu les équations d'apprentissage.
(cette partie est à completer par les nouvelles équations et l'algo proprement dit de la RBM gaussienne.)

10 Une fois entraînée, comment on utilise la RBM gaussienne ?

Après l'entraînement, on peut utiliser la RBM gaussienne pour :

- **Transformer des embeddings** : on donne un **vecteur d'embeddings** en entrée, et la RBM génère une **version transformée** des embeddings à partir des activations cachées.
- **Générer de nouvelles données** : on initialise un vecteur au hasard et on le fait passer plusieurs fois dans la RBM, ce qui produit des données similaires à celles du dataset d'entraînement.
- **Extraire des features** : on ne garde que les activations cachées h comme **représentation compacte** des données visibles.

C'est le premier cas d'utilisation qui fait l'objet de ce sujet.

11 Limites d'une RBM gaussienne

- Peut être difficile à entraîner car la variance doit être bien réglée.
- Complexité computationnelle (parallélisation recommandée)
- Moins populaire car souvent remplacée par des autoencodeurs variationnels (VAE).
- Dans le cas de données complexes (distributions complexes),la RBM Gaussienne peut s'avérer impuissante car elle suppose que les données suivent une gaussienne.

References

[1] Noam Shazeer Ashish Vaswani et al. Attention is all you need. *https://arxiv.org/abs/1706.03762*, 2023.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Yul Choi et al. Spatialvlm: Integrating spatial reasoning with vision-language models. *arXiv preprint arXiv:2105.08752*, 2021.

[4] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.

[5] Jialei Hou et al. Vila: Vision-and-language pre-training with large-scale datasets. *arXiv preprint arXiv:2103.12824*, 2021.

[6] Haesu Hwang et al. Vlm-rl: Visual-language models for reinforcement learning. *arXiv preprint arXiv:2004.06592*, 2020.

[7] Wonjae Kim et al. Vilt: Vision-and-language transformer without convolution or region supervision. *arXiv preprint arXiv:2102.03334*, 2021.

[8] Junnan Li et al. Blip: Bootstrapping language-image pre-training. *arXiv preprint arXiv:2201.12086*, 2022.

[9] Luowei Li et al. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[10] Wei Liu et al. Xmodel-vlm: Efficient cross-modal representation learning. *arXiv preprint arXiv:2104.06923*, 2021.

[11] Alec Radford et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[12] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1:194–281, 1986.

[13] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.

[14] Yao-Hung Hubert Tsai et al. Multimodal transformer for unaligned multimodal language sequences. *arXiv preprint arXiv:1906.00295*, 2019.