IA et Multimédia
Examen - 1h30

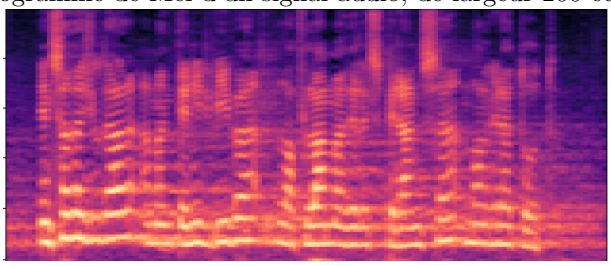
Les documents sont autorisés.

Prenez le temps de systématiquement justifier vos réponses !

Questions sur les TPs

Dans le TP1, vous avez travaillé sur la classification de langue parlée à partir d'un extrait audio, sous la forme d'un spectrogramme (Figure 1).

Figure 1: Spectrogramme de Mel d'un signal audio, de largeur 200 et de hauteur 100.



1 pt

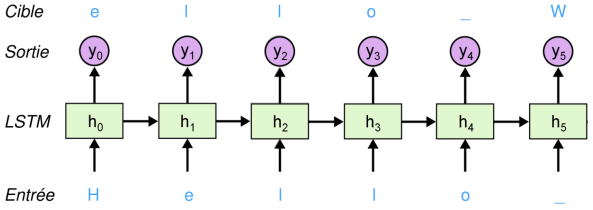
1. Diriez-vous que ce problème est de la classe *One-to-many*, *Many-to-one* ou *Many-to-many* ?
   Dans ce problème, on doit associer **une** classe (représentant la langue) à une séquence d'entrée (l'extrait audio) : ce problème est donc de la classe *Many-to-one*

2 pts

2. On décide d'utiliser une couche LSTM pour traiter ce signal. Quelle est la longueur de la séquence associée à l'image de la figure 1 ? Quelle est la taille de l'*embedding* en entrée du LSTM ?
   Le spectrogramme est un diagramme temps-fréquence : la largeur de l'image correspond donc à la longueur de la séquence (200), et la hauteur de l'image (100) correspond à la dimension de chaque élément de la séquence, qui est donc assimilable à la taille de l'*embedding* donné en entrée du LSTM.

Dans le TP2, vous avez créé un modèle de langage à l'aide d'un LSTM pour apprendre à générer des tweets à la manière de Donald Trump (Figure 2).

Figure 2: Schéma de l'architecture du modèle de langage implémenté en TP2



3. Diriez-vous que ce problème est de la classe *One-to-many*, *Many-to-one* ou *Many-to-many* ?
Sur la figure 2, on voit qu'à une séquence d'entrée ("Hello ") est associée une séquence de sortie ("ello w"). Le problème est donc de la classe *Many-to-many*. En pratique cependant, seul le dernier caractère de la séquence de sortie est important pour l'application. La réponse *Many-to-one* est donc tout à fait acceptable.

4. Lors de la génération de texte par le modèle, on utilise un paramètre de température T pour altérer les prédictions du réseau :

$$p_i = \text{softmax}(z_i^T)$$

Si $T = 1$ les prédictions du réseau ne sont pas modifiées. D'après vous, faut-il plutôt augmenter ou diminuer T pour augmenter la variabilité du texte généré par le réseau ?
Le paramètre T contrôle l'entropie des prédictions du réseau. Si T est grand, on renforce la classe de probabilité maximale et on diminue la probabilité de prédiction des autres classes (on diminue l'entropie) ; inversement, si T tend vers 0, on augmente l'entropie des prédictions ce qui tend à équilibrer les probabilités prédites pour l'ensemble des classes. Pour augmenter la variabilité du texte généré par le réseau, il faut **augmenter l'entropie** des prédictions du réseau, et donc diminuer T.

Les figures suivantes montrent des images (haut) accompagnées de leur reconstruction (bas) par un auto-encodeur construit avec des hyperparamètres variables : on fait ainsi varier l'espace latent de l'auto-encodeur (8 ou 128) et la capacité du réseau (mesurée par son nombre de paramètres : 1M ou 10M).

On a donc 4 auto-encodeurs différents :

A: Auto-encodeur de faible capacité, et à espace latent de faible dimension.

B: Auto-encodeur de forte capacité, et à espace latent de faible dimension.

C: Auto-encodeur de faible capacité, et à espace latent de haute dimension.

D: Auto-encodeur de forte capacité, et à espace latent de haute dimension.

Les figures 3, 4 et 5 ci-dessous montrent des reconstructions générées par un auto-encodeur sur l'ensemble d'apprentissage (gauche) et l'ensemble de test (droite).

Figure 3: Images originales (haut) et reconstruites (bas) de l'ensemble d'apprentissage (gauche) et de test (droite)

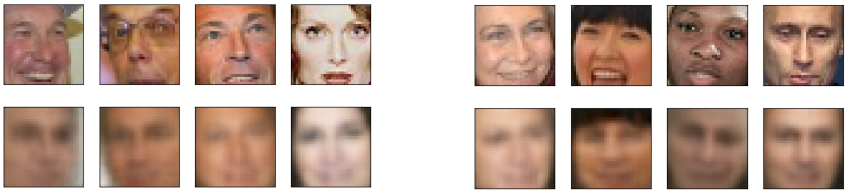


Figure 4: Images originales (haut) et reconstruites (bas) de l'ensemble d'apprentissage (gauche) et de test (droite)

Figure 5: Images originales (haut) et reconstruites (bas) de l'ensemble d'apprentissage (gauche) et de test (droite)

3 pts

5. D'après vous, quelle configuration d'auto-encodeur peut avoir produit les reconstructions des figures 3, 4 et 5 ? (**ATTENTION**, plusieurs réponses sont parfois possibles). **Justifiez votre réponse.**

   Sur la figure 3, on observe que les reconstructions de l'ensemble d'apprentissage ainsi que de l'ensemble de test sont de faible qualité. On est dans une situation de sous-apprentissage ! Ceci peut principalement s'expliquer par une capacité trop faible du réseau, et/ou par une dimension d'espace latent également un peu trop faible. On est très vraisemblablement dans la configuration A ou C.

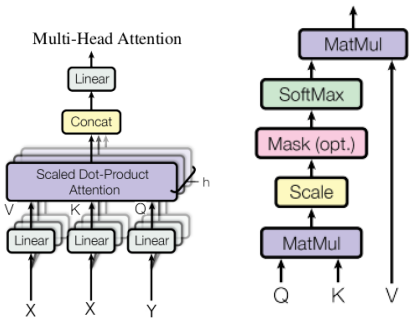
   Sur la figure 4 en revanche, tout semble bien se passer : les reconstructions sont bonnes pour l'ensemble d'apprentissage et l'ensemble de test. La configuration D est la plus probable.

   Sur la figure 5, les reconstructions de l'ensemble d'apprentissage sont excellentes, mais celles de l'ensemble de test sont mauvaises : il y a sur-apprentissage. Un cas typique de sur-apprentissage des auto-encodeurs est celui d'un espace latent trop petit combiné à une capacité très forte de l'auto-encodeur. C'est donc très probablement la configuration B qui explique ces résultats (cela pourrait aussi arriver avec la configuration D).

Exercice sur les Transformers

La figure 6 représente le détail d'une couche d'attention dans un Transformer. Ici, X pourrait correspondre à une séquence d'entrée, et Y au début d'une séquence de sortie. Les clés K, valeurs V et requêtes Q sont générées à partir des séquences X et Y, et ce pour plusieurs têtes d'attention en parallèle ; les résultats pour chaque tête sont concaténés puis repassent à travers une nouvelle couche dense.

Figure 6: Détail d'une couche d'attention multi-tête



2 pts

6. On suppose ici que la dimension des *tokens* en entrée est de 5, que la dimension interne à la couche d'attention (*embed_dim*) est de 2, et qu'il y a deux têtes d'attention en parallèle. Combien cette couche compte-t-elle de paramètres ?
   Pour chaque tête d'attention, on a 3 couches denses permettant de calculer les clés K, valeurs V et requêtes Q associées aux *tokens* des séquences d'entrée X et de sortie Y. Ces couches denses prennent en entrée un vecteur de dimension 5 et en sortie un vecteur de dimension 2 : elles comptent donc 5×2 poids synaptiques, et 2 biais. Ceci fait donc un total de $2 \times 3 \times (5 \times 2 + 2) = 72$ paramètres.

   Après l'application de l'attention, on concatène les réponses de chaque tête : on a 2 réponses de dimension 2, on obtient donc un vecteur concaténé de dimension 4. Ce vecteur est passé en entrée d'une couche dense, qui renvoie en sortie un vecteur de dimension 2. Cette dernière couche compte donc 4×2 poids synaptiques, et 2 biais, soit 10 paramètres.

   La couche d'attention compte ainsi 82 paramètres au total.

1 pt

7. Supposons que nous ayons 3 clés $K_1 = [0, 1]$, $K_2 = [1, 0]$ et $K_3 = [1, 1]$. D'après vous, sur quel élément de la séquence portera-t-on une attention maximale si on considère une requête $Q = [-0.2, 0.8]$?
   On calcule le produit scalaire entre la requête Q et les 3 clés :
   $K_1 Q^T = 0.8$, $K_2 Q^T = -0.2$, et $K_3 Q^T = 0.6$.

   On obtient le score d'attention après application d'une couche softmax à ces valeurs ; sans même la calculer, l'attention maximale sera portée sur le *token* i pour lequel le produit scalaire $K_i Q^T$ est maximal, c'est-à-dire ici le *token* 1.

   En l'occurence, ici on obtiendrait des scores d'attention de 0.46, 0.17 et 0.37 respectivement pour les *tokens* 1, 2 et 3.

1 pt

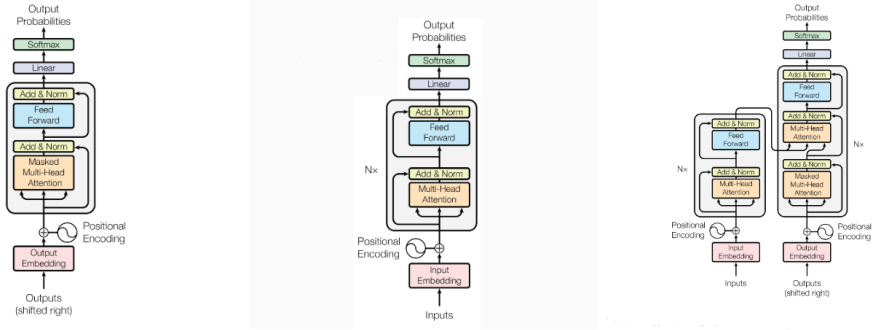
8. Comment peut-on implémenter une couche d'auto-attention à partir d'une couche d'attention ?
   Il suffit d'appeler la couche d'attention représentée figure 6 en indiquant la même séquence comme séquence d'entrée (X sur le schéma) et de sortie (Y sur le schéma).

La figure 7 présente 3 variantes d'architecture de *Transformer*.

1 pt

9. Nous avons étudié en cours 3 architectures : celle de l'article *Attention is all you need*, celle de `GPT` et celle de `BERT`. Retrouvez les correspondances entre la figure 7 et ces 3 architectures.

Figure 7: Trois architectures de Transformers différentes :



La 3e architecture est l'architecture du premier *Transformer*, celle de l'article *Attention is all you need*. La 1e architecture correspond au décodeur (partie droite) du *Transformer* originel, c'est une architecture de type `GPT`. La 2e architecture correspond à l'encodeur (partie gauche) du *Transformer* originel, c'est une architecture de type `BERT`. La différence entre ces 2 architectures tient à la première couche d'auto-attention, qui est masquée dans le cas de `GPT`, et qui ne l'est pas dans le cas de `BERT`.

1 pt

10. Il est possible de pré-entraîner certaines ces architectures de manière non supervisée en fournissant des entrées masquées à compléter. Retrouvez, parmi ces deux types d'entrées masquées, laquelle correspond à quelle architecture:

    - To be or not to [mask], that is the question.
    - To be or not to be, that is the [mask].

    Expliquez votre réponse.
    Lorsque le mot masqué se trouve au milieu de la phrase, on peut choisir d'utiliser les mots précédents mais aussi les mots suivants pour prédire le mot masqué. Ceci nécessite une architecture dans laquelle la couche d'auto-attention est non masquée, c'est-à-dire celle de `BERT` (le **B** signifie d'ailleurs Bidirectionnel).

    Inversement, si le mot masqué est situé à la fin de la phrase, cela correspond à un scénario où l'on veut prédire le prochain mot ; il faut donc masquer la couche d'auto-attention pour éviter de prendre en compte les mots suivants (qui, en fait, sont masqués également). Cela correspond à l'architecture `GPT`.

1 pt

11. Voici 3 tâches classiques en traitement du langage naturel : classification de texte (par exemple: classification d'un email en spam/non spam), génération de texte, et traduction automatique. D'après vous, quelle architecture est la plus adaptée à chacune de ces tâches ? (justifiez votre réponse)
    La traduction automatique est une tâche qui nécessite une architecture encodeur-décodeur, avec une séquence d'entrée (la phrase à traduire) et une séquence de sortie (le début de la phrase traduite) : il faut donc utiliser l'architecture de *Transformer* de l'article *Attention is all you need*.

    Comme on l'a dit précédemment, la génération de texte est une application dans laquelle on veut prédire le prochain mot : il faut donc plutôt utiliser `GPT`.

    Enfin, la classification de texte tirera parti d'une analyse complète, bidirectionnelle, de la séquence d'entrée ; c'est la raison pour laquelle `BERT` est bien adaptée à ce type de problèmes.

2 pts

12. Pour chacune des 3 tâches décrites dans la question précédente, décrivez le format des entrées et des sorties, ainsi que le choix de la couche de sortie que vous feriez (nombre de neurones et fonction d'activation).

(a) **Classification de texte :** Le réseau prend en entrée une séquence de *tokens* et renvoit en sortie des probabilités de classe, en fonction du contexte. Si l'on prend le problème spam/non spam (classification binaire), on aura un unique neurone en sortie avec une activation sigmoide.

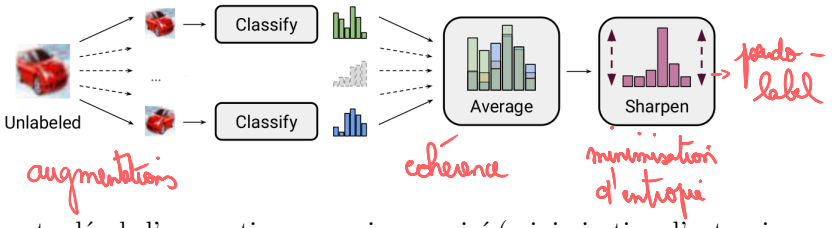
(b) **Génération de texte :** Le réseau prend en entrée une séquence de *tokens* et renvoit en sortie des probabilités de classe, une pour chaque *token*. On aura donc autant de neurones en sortie qu'il y a de *tokens* dans le vocabulaire (*vocab_size*) et une activation `softmax`.

(c) **Traduction automatique** Le réseau prend en entrée une séquence de *tokens* et renvoit en sortie des probabilités de classe, une pour chaque *token*. On aura donc autant de neurones en sortie qu'il y a de *tokens* dans le vocabulaire (*vocab_size*) et une activation `softmax`.

Question sur la faible supervision

La figure 8 rappelle les éléments du pipeline de l'algorithme MixMatch.

Figure 8: Schéma de l'algorithme MixMatch



3 pts

13. Parmi les concepts clés de l'apprentissage semi-supervisé (minimisation d'entropie, pseudo-labels, coût de cohérence, augmentation de donnée), lesquels sont utilisés dans MixMatch ? Positionnez les sur la figure et expliquez en quelques mots votre réponse.
Dans `MixMatch`, on commence par générer K **augmentations** d'une image non labellisée. On calcule la prédiction du modèle sur ces K augmentations puis on fusionne ces prédictions ; à partir de là, on traitera toutes les augmentations de la même manière, ce qui est une manière d'assurer une **cohérence** entre les différentes augmentations. L'opération de *sharpening* diminue l'entropie de la distribution de probabilités associée à la donnée non labellisée (**minimisation d'entropie**), et le résultat constitue un **pseudo-label** qui sera utilisé pour l'entraînement.