DUJARDIN Thomas

ENSAE 2e année

Stage d'application

Année 2021-2022

Deep Learning pour la reconnaissance multimodale d'émotions dans des extraits vidéos

MICS-IECL
Gif-sur-Yvette, 91190

Marianne CLAUSEL et Myriam TAMI
Du 01/06/2022 au 30/09/2022

Remerciements

Je tiens à remercier M. Nicolas CHOPIN pour m'avoir proposé ce stage qui fut fort intéressant et pertinent dans le cadre de mon parcours scolaire.

Je remercie également Mmes Myriam TAMI du laboratoire MICS, et Marianne Clausel de l'Institut Elie-Cartan de Lorraine pour leur encadrement tout au long de ce stage, ainsi que M. Romain MARTIN de Two-I pour son appui et son assistance.

Je remercie enfin le laboratoire MICS et son personnel pour leur accueil chaleureux qui a su rendre agréables ces trois mois de stage.

Introduction

Cette année, j'ai choisi d'effectuer un stage d'application de trois mois au sein du laboratoire Mathematics and InformatiCS (MICS) de CentraleSupélec, sur une problématique centrée autour du deep learning appliqué à la détection d'émotions dans les conversations. Ce choix est avant tout motivé par ma volonté d'approfondir mes connaissances en deep learning appliqué à des modalités qui ne sont pas habituellement étudiées à l'ENSAE (vidéo, audio, …).

Mon sujet s'inscrit dans le cadre de travaux entamés par une doctorante en thèse CIFRE chez Two-I, en psychologie appliquée à l'intelligence artificielle, sur de la détection d'émotions appliquée au marketing/à la santé. Two-I est une start-up qui traite ce type de sujets, et qui, plus généralement, produit des solutions logicielles d'analyse vidéo. Cette doctorante était confrontée à de nombreuses problématiques dans ces domaines : automatisation de l'évaluation de la satisfaction client dans les surfaces commerciales, élaboration d'outils de détection faciale de la douleur (afin d'éviter les capteurs chez les grands brûlés, entre autres), évaluation de l'émotion prédominante à chaque instant d'une conversation entre deux locuteurs… Celles-ci l'ont amenée à considérer des algorithmes de détection d'émotions, capables d'interpréter des données dites « multimodales ». Les données « multimodales » sont des données qui proviennent de plusieurs modalités, c'est-à-dire qu'elles proviennent de plusieurs types de capteurs (appareil photo, thermomètre, microphone, caméra, …), qui mesurent chacun un type de signal (image, température, audio, vidéo, …). On cherche alors, dans les algorithmes « multimodaux », à utiliser toutes ces modalités à la fois afin de produire un résultat, que ce soit une dashboard de monitoring en temps réel, une décision… Dans le cadre de ce stage, je me suis concentré sur la détection d'émotions dans les conversations. Les algorithmes de détection d'émotions multimodale étaient entraînés sur les modalités texte, vidéo et audio, ce qui pose deux problèmes importants qui seront détaillés dans ce rapport : le problème de l'alignement du texte avec la vidéo, afin de produire des bases de données cohérentes et exploitables, et le problème de l'association d'un visage, d'un audio et de sa transcription à un locuteur. L'objet de ce stage était donc d'étudier les méthodes de prédiction de l'émotion prédominante dans des datasets multimodaux de conversations entre deux locuteurs, ce qui m'a amené à considérer les problématiques de la détection d'émotions, et de la gestion de la multimodalité des données.

Au cours de ma deuxième année à l'ENSAE, en préparation d'un tel stage et de ma troisième année, j'ai effectué la MOOC « Deep Learning Specialization » de Andrew Ng sur Coursera, qui m'a enseigné les réseaux de neurones dans le détail, la gestion de l'entraînement d'un modèle, les réseaux de neurones convolutifs, les transformers… Cela m'a beaucoup facilité le travail pour le début de ce stage, en réduisant fortement la période de familiarisation avec le deep learning, prévue initialement pour durer au moins deux semaines.

Suite à cela, j'ai été amené à effectuer une revue de littérature en analyse d'émotions par deep learning, que j'ai exposée à mes encadrantes dans plusieurs présentations. J'ai ainsi pu me familiariser avec les deux grandes étapes de tout modèle d'analyse d'émotions multimodale : en premier lieu, puisque les données ne sont pas toutes de la même nature, il y a nécessité de featuriser chaque modalité de données, c'est-à-dire de les faire passer dans un modèle pour en extraire des vecteurs représentatifs que les modèles classiques de deep learning peuvent comprendre. En second lieu, il convient de trouver une manière de fusionner les vecteurs précédemment obtenus, afin d'obtenir des features représentant toutes les modalités à la fois, et ce avec une certaine mémoire des données passées. Ce qui justifie l'utilisation de données multimodales dans le cadre de ce problème plutôt que simplement la transcription du dialogue, c'est le fait que les modalités audio et vidéo apportent de l'information et affinent la prédiction d'une émotion : lorsque l'on utilise toutes les modalités disponibles, que ce soit séparément (un modèle chacun, et la décision finale est une combinaison des décisions des modèles) ou conjointement (un modèle commun avec fusion des modalités), il est possible que les informations issues de chaque modalité se contredisent, ce qui peut permettre d'accéder à plus d'informations sur l'émotion présente. Par exemple, si une phrase prononcée est perçue comme exprimant de la joie, alors que le visage de la personne la prononçant exprime de l'agacement, la contradiction entre modalités ainsi obtenue pourrait signifier que la personne est ironique, chose qu'il n'aurait pas été possible de détecter à l'aide d'une seule de ces modalités. La dynamique de la conversation est également exploitable pour affiner une prédiction : le contexte global du dialogue peut permettre d'ajouter de l'information lorsque l'on traite une phrase précise : savoir qu'un locuteur est principalement en colère dans un dialogue peut permettre de corriger une prédiction « neutre » en une prédiction « colère » / « agacement ». Une fois les vecteurs de features fusionnés obtenus, la dynamique de la conversation peut se modéliser en leur appliquant des modèles récurrents comme les RNN, ou bien des modèles à mécanismes d'attention capables de tenir compte d'un historique dans des données avec un ordre temporel. Dans les modèles de détection d'émotions dans les

conversations les plus récents, ce sont ces derniers qui sont utilisés pour modéliser cette dynamique.

J'ai ensuite pu implémenter un tel modèle avec Python, ce qui m'a permis de me familiariser avec le framework de machine learning PyTorch, et les centres de calcul.

Ce rapport est composé de trois grandes parties : dans la première, j'effectue une revue de littérature dans le domaine de la détection multimodale d'émotions, c'est-à-dire à la fois dans les domaines de la fusion de modalités et de la featurization du son, du texte et de l'image. Dans la seconde partie, je rentre dans les détails théoriques de mon modèle de détection d'émotions, fortement inspiré du modèle « M2FNet » [12]. Enfin, dans une dernière partie, je développe sur l'entraînement de mon modèle : les hyperparamètres choisis, l'architecture de mon programme Python, ses performances et défauts…

Le sommaire est cliquable et permet de naviguer facilement dans le rapport. Une bibliographie est disponible en fin de rapport.

Table des matières

I) Le problème : apprentissage statistique sur des données multimodales appliqué à la détection d'émotions……p.7

I.1) La problématique considérée

I.2) Revue de littérature

I.2.1) Extracteur de features

I.2.2) Fusion de modalités

I.2.3) Les modèles de classification multimodale d'émotions

II) La classification multimodale d'émotions : le modèle M2FNet……p.10

II.1) Les bases de tout algorithme moderne de détection d'émotions

II.1.1) Comment tenir compte de la multimodalité des données ?

II.1.2) Les transformers encoders

II.2) Featurization des différentes modalités

II.2.1) Détection des « Action Units »

II.2.2) Vectorisation du texte

II.2.3) Vectorisation du son

II.3) L'architecture du modèle MF2Net

III) Implémentation, entraînement et performances du modèle……p.22

III.1) L'implémentation de MF2Net

III.2) L'entraînement du modèle

III.3) Résultats

IV) Conclusion……p.26

Bibliographie……p.27

I) Le problème : apprentissage statistique sur des données multimodales appliqué à la détection d'émotions

I.1) La problématique considérée

Tout l'objet de mon stage était de déterminer, à partir de bases de données consistant en des vidéos de dialogue entre deux locuteurs accompagnées de la piste audio et d'une transcription de celle-ci, l'émotion prédominante d'un locuteur à un instant t, en utilisant à la fois les informations apportées par chaque modalité à l'instant t, et l'historique des informations extraites aux instants précédents $s < t$. L'élaboration d'un modèle de Deep Learning multimodal pour la détection d'émotions sur des extraits vidéos nécessite alors en premier lieu des extracteurs de features efficaces pour chaque modalité, et, en second lieu, un module de fusion de modalités capable de produire un vecteur final qui est représentatif de toutes les modalités de données, et qui possède une notion de « contexte » pour tenir compte de la dynamique d'une conversation. Durant le début de mon stage, j'ai pris le temps de faire une revue de tous les modèles utiles à ces fins, et de tous les modèles de reconnaissance d'émotions dans les conversations.

I.2) Revue de littérature

Je me suis donc intéressé aux états de l'art en termes d'extracteurs de features, de fusion de modalités, et de modèles de détections multimodale d'émotions dans les conversations.

I.2.1) Extracteurs de features

a) Features visuelles

Elles sont de deux types : les premières sont celles que l'on extrait directement sur le visage, les secondes sont celles qui proviennent du reste de la vidéo, qui n'est pas toujours neutre et peut apporter des informations, comme la gestuelle d'un locuteur. Ici, on ne considère que les premières. La lecture des émotions d'un visage passe par les « Actions Units », mouvements élémentaires des muscles constitutifs des émotions, que l'on cherche à inférer.

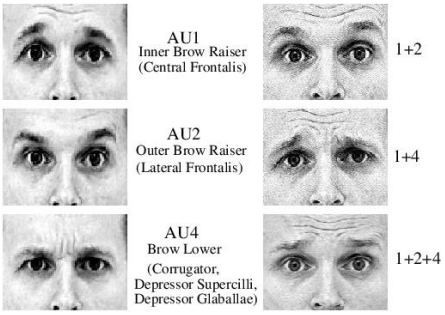


Figure 1 : Quelques actions units (crédits : Javier R. Movellan, ResearchGate)

Une action unit (« A.U. ») est la contraction ou la détente d'un ou plusieurs muscles du visage, responsable d'un mouvement précis de celui-ci. Par exemple l'AU1, Inner Brow Raiser, correspond au fait de lever les sourcils. Ces action units sont constitutives d'une nomenclature nommée *Facial Action Coding System (FACS)* [18] créée par Paul Ekman et Wallace Friesen dans les années 70 et qui, aujourd'hui encore, est un des outils de description principaux dans la lecture des expressions faciales, puisqu'il permet de décrire aisément les émotions comme des combinaisons d'AUs [26] (ce qui s'écrit par exemple Happiness = AU 1, 6, 12, 13). Ces AUs peuvent être accompagnées d'une intensité qui s'échelonne en cinq niveaux, de « AU absente » (0) à « AU très intense » (4). Ces actions units se récupèrent sur chaque image de la vidéo, à la suite d'une extraction du ou des visages présents dessus.

Un premier modèle [1] détecte ces Actions Units et leur intensité en produisant une représentation vectorisée d'un visage qui repose sur des gradients discrets et sur les positions de soixante-six marqueurs sur ce visage, et qui envoie cette représentation dans un SVM. Ce premier modèle est efficace et simple à implémenter, mais un second modèle, plus récent et qui repose sur des Transformers et des CNN, [2] atteint l'état de l'art pour la plupart des Actions Units.

b) Features textuelles

Il y a tout d'abord le très basique *one-hot encoding* pour la vectorisation du texte, qui est très simple à implémenter, mais qui ne permet pas tenir compte des relations entre les mots d'une phrase. On trouve ensuite des modèles plus élaborés, de type *bag-of-words*, qui s'appuient sur des décomptes (fréquences de mots, n-grams, TF-IDF…) afin de produire des vecteurs représentatifs. On a également les approches capables d'apprendre des relations locales mais non globales par réseaux de neurones, comme Word2Vec (2013) [3] qui encode un mot par les poids d'un réseau de neurones à une couche cachée. Ces deux approches sont réunies dans GloVe (2014) [4], qui génère des vecteurs pour chaque mot par apprentissage statistique, avec

une fonction de perte qui dépend de l'analyse statistique de corpus de textes d'entraînement. Les réseaux de neurones récurrents (type LSTM bidirectionnels) ont ensuite pris le pas sur ces modèles dans les benchmarks d'évaluation, en permettant une bien meilleure gestion du contexte d'une phrase, et des liens entre les mots de celle-ci. Enfin, les modèles les plus récents et les plus performants sur des tâches de classification de texte/de vectorisation de texte utilisent des *Transformers* (BERT (2018) [5] et ses variantes), parfois utilisés avec des *Graph Convolutional Networks* [6], ou encore des modèles du type XLNet qui pallient certains défauts des BERT [7]. Les modèles à base de transformers, comme BERT, constituent une révolution dans le domaine de la NLP. Ces derniers, bien plus performants que ceux précédemment cités, ont une bien meilleure capacité à produire des représentations vectorisées de phrases, et, surtout, tiennent mieux compte du contexte global d'une conversation grâce aux *attention modules* [13]. La partie II.1.2 de ce rapport détaille les architectures transformers, et les compare aux LSTM et autres RNN sur des tâches de classification de texte.

c) Features audio

Puisque l'on traite de dialogues entre deux locuteurs, il est nécessaire d'utiliser un modèle spécialisé dans le traitement de la parole. Je me suis intéressé particulièrement à deux modèles de Meta AI, Wav2Vec2.0 (2020) [8] et HuBERT (2021) [9], que j'ai vu employés dans différents papiers de reconnaissance d'émotions, et dont les fonctionnements, qui sont très similaires, reposent sur les Transformers. HuBERT fait usage des « MFCC » [27], coefficients extraits par une succession d'opérations mathématiques reposant largement sur la transformée de Fourier et sur la DCT, constituent déjà des features intéressantes pour le son.

I.2.2) Fusion de modalités

Une fois ces vecteurs obtenus pour chaque modalité, il est nécessaire d'en tirer d'autres vecteurs qui répondent à la problématique de classification multimodale d'émotions, et qui mêlent donc des éléments venant de toutes les modalités. Il existe pour cela deux principales stratégies, décrites ci-dessous.

a) La simple concaténation

La manière la plus simple et la plus intuitive de former des vecteurs représentatifs de plusieurs modalités à la fois consiste simplement à concaténer les vecteurs de features issues

de chacune de ces modalités. Cela peut se faire de trois manières différentes, décrites sur la figure ci-dessous. La plus efficace d'entre elles semble être la joint fusion [10].

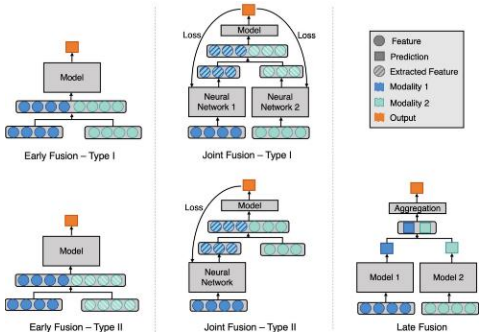


Figure 2 : trois types de concaténation (source : Huang, SC., Pareek, A., Seyyedi, S. et al. Fusion of medical imaging and electronic health records using deep learning: a systematic review and implementation guidelines.)

b) La fusion à base de Transformers

Les modèles récents de classification multimodale d'émotions font usage de Transformers afin de fusionner les modalités. En effet, grâce à leurs trois entrées et au mécanisme d'attention, ceux-ci permettent de produire des vecteurs bien mieux représentatifs de plusieurs modalités à la fois en vue d'une classification multimodale qu'une simple concaténation. Ceux-ci sont détaillés dans la partie II.1.2 de ce rapport.

I.2.3) – Les modèles de classification multimodale d'émotions

En matière de reconnaissance multimodale d'émotions dans les conversations, je me suis intéressé aux datasets MELD (Multimodal EmotionLines Dataset, décrit en partie II) et IEMOCAP (Interactive Emotional Dyadic Motion Capture), dataset de 12 h de vidéos de conversations labelisées dans lesquelles des acteurs jouent des mises en scène, avec des points sur le visage afin de les isoler aisément. Sur IEMOCAP, c'est EmoCaps (2022) [11] qui domine le classement. Après diverses extractions de features, ce modèle fait usage de transformers pour fusionner les modalités, puis utilise un bloc de LSTM bidirectionnels afin de tenir compte du contexte global de la conversation, nécessaire à la détermination de l'émotion à un instant donné. En revanche, sur MELD, EmoCaps est bien moins performant que M2FNet [12]. Ce deuxième modèle emploie des transformers à la fois pour fusionner les modalités et pour modéliser le contexte à travers la conversation. M2FNet est, par ailleurs, le deuxième modèle le plus performant sur IEMOCAP.

II) La classification multimodale d'émotions : le modèle M2FNet

Je me suis donc attelé à l'élaboration d'un modèle de reconnaissance d'émotions dans les conversations. J'ai décidé en ce sens de m'inspirer fortement du modèle M2FNet [12], qui réalise l'état de l'art sur le dataset MELD en termes de score F1 pondéré. Ce dataset, illustré ci-dessous, consiste en 1 400 dialogues découpés en 13 000 utterances de quelques secondes chacune, avec en moyenne trois émotions exprimées par dialogue. Ces dialogues ont lieu entre deux personnages issus de la sitcom Friends, et leurs utterances sont labellisées par sentiment (positif, neutre, négatif) et par émotion (tristesse, colère, …). Il y a une assez grande variété de locuteurs présents dans ce dataset : 260 dans l'ensemble de train, 46 dans celui de dev et 100 dans celui de test. Ce sont alors les modalités texte, audio et vidéo que le modèle traite.

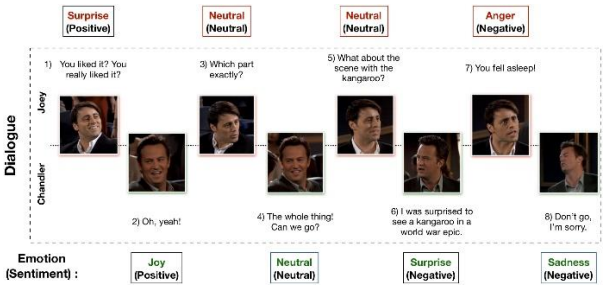


Figure 3 : Illustration du dataset "MELD"

Ici, le problème de l'alignement de la transcription du dialogue avec la vidéo (cité en introduction) est résolu directement par MELD, puisque celui-ci découpe le dialogue en « utterances » qui correspondent chacun à une prise de parole continue d'un des deux locuteurs, comme cela est illustré sur la figure 3. En revanche, le deuxième problème évoqué en introduction (l'assignation d'un locuteur à une phrase transcrite) est nettement plus complexe, surtout dans un dataset comme celui-ci où les extraits peuvent contenir deux visages ou plus, dont certains n'appartiennent à aucun des locuteurs. Dans ce cas, on peut simplement effectuer une sorte de moyenne pondérée des features extraites sur les visages détectés, ce qui se justifie par le fait que, dans une « utterance », le visage de la personne qui ne parle pas, s'il est présent dans la vidéo, peut aussi apporter de l'information sur l'émotion de celui du locuteur. Cela empêche d'avoir à recourir à des solutions de reconnaissance faciale comme YOLO afin d'identifier le personnage de Friends qui parle, ce qui rendrait le modèle impossible à généraliser à des datasets non issus de Friends.

II.1) – Les bases de tout algorithme moderne de détection d'émotions

II.1.1) – Comment tenir compte de la multimodalité des données ?

Comme cela est expliqué dans l'introduction de ce rapport, tous les modèles que j'ai pu étudier durant ce stage procèdent en deux temps afin d'incorporer des éléments de toutes les modalités à l'apprentissage : dans un premier temps, ceux-ci featurizent les différentes modalités et leur apporte d'éventuels traitements supplémentaires (prise en compte de la dynamique de la conversation par exemple), puis dans un second temps, ils fusionnent les vecteurs de features. Chaque modèle aborde ces deux étapes avec des algorithmes différents. Je détaillerai ci-dessous les différentes briques d'un modèle de détection multimodale d'émotions, de la featurisation des modalités à leur fusion, avant d'exposer le modèle M2FNet [12] que j'ai utilisé, avec les modifications que j'y ai apporté.

II.1.2) – les transformers encoders

a) – détails de l'architecture

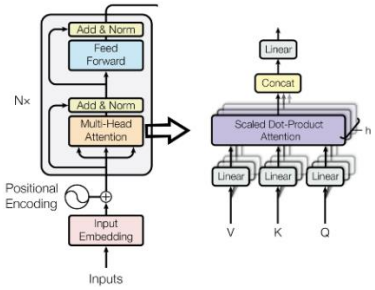


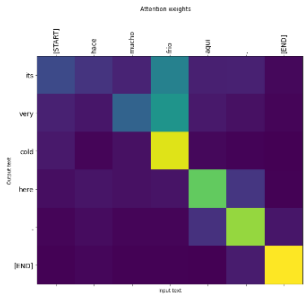
Figure 4 : à gauche l'encoder d'un transformer, à droite l'architecture de la multihead attention (crédits : factored.ai et paperswithcode.com)

Le modèle de détection multimodale d'émotions implémenté durant mon stage repose en grande partie sur les Transformers Encoders de Vaswani et al. (2017) [13], qui sont assez intuitifs. Avant d'être passé dans l'encoder (bloc gris à gauche), un input (une phrase dans le modèle de base, mais cela peut être généralisé à d'autres modalités) est « tokenizé », c'est-à-dire qu'il est transformé en un vecteur d'embeddings de taille fixe d (souvent 512), et qu'on lui ajoute un *Positional Encoding* (P.E.) [13]. L'intérêt du PE réside dans le fait que les transformers ne contiennent ni récurrence, ni convolution, ce qui rend nécessaire l'ajout d'une information sur la position de chaque mot dans la phrase, sous la forme d'un vecteur PE de la forme :

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Où pos est la position du mot dans la phrase, et i la i-e coordonnée du vecteur. Puisque PE_{pos+k} est une fonction linéaire de PE_{pos}, cet ajout permet d'injecter une information sur la position de chaque mot (ou chaque coordonnée) dans le vecteur d'embeddings.

Ensuite, le vecteur tokenizé est passé dans un multi-head attention, qui résulte de la concaténation de plusieurs modules d'attention. Un module d'attention permet d'étudier les corrélations entre les mots de deux phrases, et prend trois entrées : la Key K, la Query Q et la Value V.



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

Figure 5 : à gauche, diagramme de l'Attention entre une phrase et sa traduction, à droite, formule donnant l'Attention. (crédits : tensorflow.org)

C'est ici un module d'auto-attention que l'on utilise, pour déterminer les liens entre les mots d'une seule phrase. La Key, la Query et la Value sont alors trois fois la même phrase. L'intérêt de la multi-head attention dans ce modèle réside alors dans le fait qu'un seul module d'attention pourrait occasionner une représentation biaisée de l'input. L'association de plusieurs de ces modules permet de produire plusieurs représentations de l'input, qui sont ensuite concaténées puis passées dans un filtre linéaire.

La couche suivante, « Add & Norm », se décompose en deux temps : dans un premier temps, afin de stabiliser numériquement les sorties d'une composante, on procède à une *Layer Normalization*, normalisation à l'échelle d'une couche plutôt qu'à l'échelle d'un batch. Dans un deuxième temps, afin de mieux propager le gradient dans l'encoder du transformer, on ajoute des « residual skip connections » autour de chaque composante, ce qui consiste en l'ajout de l'identité à l'output de celle-ci, permettant à chacune d'entre elles de modéliser la fonction identité et de limiter leur potentiel impact négatif. La dernière composante est un réseau de neurones avec une couche cachée et une activation SwiGLU (plus performante qu'une ReLU [14]), suivie d'un bloc « Add & Norm » comme précédemment.

Le bloc d'encoder est alors répété plusieurs fois, et la sortie du dernier bloc est la représentation vectorisée finale de l'input.

b) – Comparaison des transformers avec d'autres modèles

Puisque les données traitées (le dataset MELD) ont une composante temporelle, il m'a semblé nécessaire d'utiliser des architectures capables d'en tenir compte. En ce sens, je me suis intéressé aux réseaux de neurones récurrents (LSTM, GRU, …) et aux transformers, que j'ai comparés sur des tâches similaires à la problématique de mon stage, en termes de performances et de rapidité, grâce à la plateforme Paperswithcode de Meta.

Concernant les tâches de reconnaissance d'émotion multimodale dans les dialogues, ce sont les transformers qui dominent le classement sur le dataset MELD (suivis de près par des GNN). Concernant la reconnaissance d'émotion multimodale, c'est le modèle *COGMEN* [15] mêlant GNN et transformers qui domine assez largement le classement. Sur le dataset CMU-MOSEI de reconnaissance d'émotion multimodale portant sur 23 500 phrases extraites de vidéos YouTube d'un millier d'utilisateurs, transformers et LSTM réalisent tous les deux l'état de l'art en termes d'accuracy. En termes de nombre de paramètres et de vitesse, des papiers [16, 17] mettent en évidence des vitesses de training et d'inférences beaucoup plus grandes pour des transformers que pour des modèles LSTM bidirectionnels avec nettement moins de paramètres, sur des tâches de classification de textes et de reconnaissance de voix.

II.2) – Featurization des différentes modalités

Ainsi qu'expliqué précédemment, il est nécessaire de vectoriser (ou « featuriser ») chacune des modalités, afin de produire des représentations numériques aisément fusionnables dans le cadre d'un apprentissage multimodal. Cette sous-partie détaille le procédé de featurization pour chacune des modalités utilisées. Les modèles de featurisation décrits ci-dessous sont ceux que j'ai utilisés lors de ma propre implémentation du modèle M2FNet, et sont différents mais proches de ceux qui ont été utilisés par les auteurs du papier. Pour la détection des visages, j'utilise la librairie py-feat de Python qui implémente le papier [1], et les auteurs utilisent un réseau de neurones convolutionnel pour détecter les visages et en extraire des features. Ils incorporent également des éléments issus du reste de la vidéo, chose que je ne fais pas. Quant à la featurisation du texte, les auteurs du papier utilisent un RoBERTa, variante de BERT [5], assez proche du modèle BART que j'utilise. Enfin, pour featuriser la piste audio, les auteurs utilisent leur propre modèle d'extraction de features s'appuyant sur le

Mel Spectrogram [27], qui n'est décrit que sommairement dans le papier, et j'utilise le Wav2Vec2.0 [8] de Meta AI. Contrairement à M2FNet, dont les extracteurs de features sont ré-entraînés par une triplet loss pour les renforcer, mes extracteurs de features sont des versions pré-entraînées.

II.2.1) – détection des « Action Units »

a) – passage par les action units – pourquoi et comment

Les bases de données disponibles en ligne ou sur demande pour la détection multimodale d'émotions consistent en des vidéos dont le fond n'est pas toujours neutre. Un outil de détection de visages puis de détection d'émotions sur ceux-ci permet de ne se concentrer que sur ce qui détermine le mieux l'émotion dans la vidéo. Comme cela est expliqué dans la partie I.2.1.a) « Features visuelles », on passe par les Action Units pour déterminer l'émotion sur un visage.

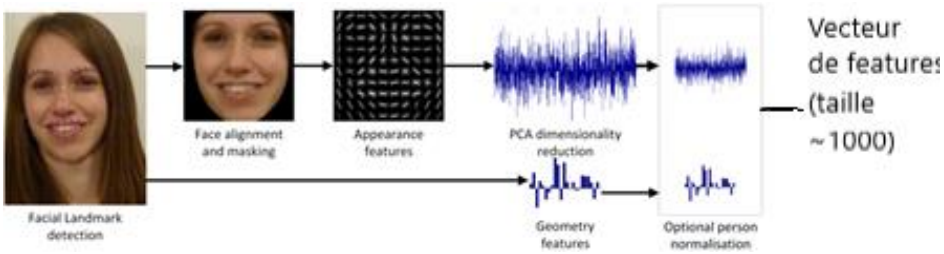


Figure 6 : vectorisation d'un visage (source : papier de Baltrusaitis et al. [1])

Pour récupérer un vecteur représentatif d'un visage en passant par les AUs, j'ai utilisé la librairie Python « py-feat », spécialisée dans la détection des visages et dans la lecture de leurs émotions. Le module de détection d'AUs de cette librairie repose sur le papier de Baltrusaitis et al. [19], dont l'algorithme est décrit dans la figure 6. On y voit qu'avant toute featurization, la première étape consiste en l'isolement du visage sur l'image.

b) – détection du visage

C'est le modèle RetinaFace [25] qui est utilisé pour détecter les visages dans chaque frame des vidéos. Ce modèle permet une détection single-shot de visages sur n'importe quel fond, en apprenant à prédire des *bounding boxes* par un réseau CNN. Plus précisément, le réseau apprend à inférer la position du coin supérieur gauche et les dimensions des bounding boxes. Les visages dans les bounding boxes ainsi prédites sont ensuite isolés, puis analysés.

c) – isolement du visage

Le papier de Baltrusaitis et al. [1] isole le visage à l'aide de 66 *facial features*, points saillants du visage (coin des lèvres, pointe du nez, …), lesquelles sont détectées grâce à un modèle de la famille des *Constrained Local Model* [19]. Ce modèle effectue un fitting des facial features sur le visage à l'aide d'une base de données d'images de visages labellisées et prises à partir de la même vue. A partir de la position moyenne de la i-e facial feature dans la base de données, l'algorithme cherche la meilleure transformation pour envoyer le point moyen $\bar{x_i}$ vers le point x_i sur le visage non labellisé. Cette transformation a une composante rigide (rotation R_{2D}, scaling s et translation t) et non-rigide (composante $\phi_i q$).

$$x_i = s.R_{2D}.(\bar{x_i} + \Phi_i q) + \mathbf{t}$$

Figure 7 : fitting d'une transformation pour récupérer les facial landmarks ([19]).

Les composantes rigides et non-rigides de cette transformation sont consignées dans un vecteur p = [s, R_{2D}, t, q], et le problème de fitting pour obtenir $p_{optimal}$ consiste, à partir d'une estimation p_0 du vecteur $p_{optimal}$, à chercher Δp tel que $p_0 + \Delta p = p_{optimal}$ en résolvant :

$$\arg\min_{\Delta \mathbf{p}} \left[\mathcal{R}(\mathbf{p}_0 + \Delta \mathbf{p}) + \sum_{i=1}^{n} \mathcal{D}_i(\mathbf{x}_i; \mathcal{I}) \right]$$

Figure 8 : problème de minimisation du CLM ([19])

Où R est un terme de régularisation qui pénalise les transformations trop complexes ou improbables, et D un terme qui représente le défaut d'alignement de la facial feature i au point x_i en l'image I. Dans les faits, c'est un problème légèrement différent qu'on résout (appelé le *Regularised Landmark Mean Shift*), qui est équivalent à celui-ci [19].

Une fois les facial features récupérées, on se sert des plus stables (i. e. celles qui n'ont pas de variations spatiales trop importantes par rapport à leurs positions sur un visage neutre, comme celles de la bouche) pour isoler le visage. Pour ce faire, on commence par projeter le visage sur un espace 112x112 avec une distance interoculaire fixée (pour avoir une même vue pour tous les visages de la base de données), on élève légèrement les facial features du haut du visage afin de récupérer les rides du front qui ont un intérêt dans la détection d'émotions, puis on récupère l'enveloppe convexe de la forme géométrique formée par ces facial features.

Figure 9 : isolement du visage par CLM

d) – extraction des features du visage

Ainsi que l'illustre la figure 6, il y a deux sortes de features. Les premières, les *geometry features*, consistent simplement en les coefficients du vecteur $\phi_i q$. Les secondes, les *appereance features*, utilisent des *Histograms of Oriented Gradients (HOG)* [20]. Il s'agit de réduire une image en un vecteur de features à l'aide de gradients discrets. Pour cela, on subdivise l'image en blocs réguliers 8x8, on la grayscale, et on calcule, pour chaque pixel de chaque bloc, la norme et l'orientation (de 0 à 180°) du gradient discret. Chaque bloc est représenté par un vecteur de 9 coefficients (correspondant à 9 intervalles de 20° entre 0 et 180°). Pour un pixel donné, on place la norme de son gradient dans la case (ou les cases) du vecteur correspondant à son orientation. En sommant les contributions de chaque pixel, on obtient ce vecteur.

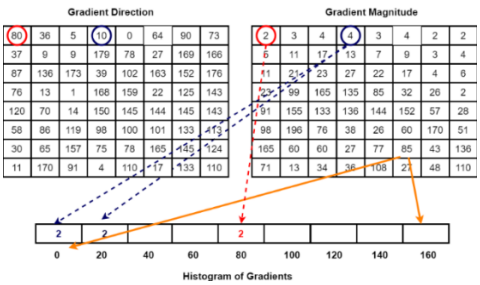


Figure 10 : création du vecteur des HOG pour un bloc (crédits : Mrinal Tyagi, towardsdatascience)

Figure 11 : le carré bleu (2x2 blocs de taille 8x8) traverse l'image avec une stride de 1

Mais en l'état, les HOGs sont sensibles aux variations de luminosité au sein de l'image, plutôt qu'aux variations intéressantes de l'image. Pour résoudre ce problème, on procède à une normalisation par blocs 2x2 avec une stride de 1 (cf. figure 11) : les vecteurs issus des 4 blocs

contenus dans le carré bleu sont concaténés en un vecteur qui est normalisé (L2). Le vecteur représentatif obtenu résulte alors de la concaténation des vecteurs normalisés ainsi.

Ce vecteur étant particulièrement volumineux (plus de 4 000 entrées), on en réduit la taille d'un facteur 4 par ACP. Le vecteur représentatif final obtenu résulte de la concaténation de ce vecteur obtenu par ACP, et de celui des *geometry features*, et a une longueur d'environ 1 600.

e) – normalisation des visages

Les auteurs du papier [1] ont constaté que tous les individus présents sur les datasets qu'ils ont utilisés n'ont pas la même expression faciale au repos : par exemple, certains froncent naturellement plus les sourcils que d'autres. Pour tenir compte de cette disparité, les auteurs ont simplement calculé la médiane (temporelle, à travers la vidéo) des features extraites, et l'ont soustraite au vecteur de features du visage, en faisant l'hypothèse que l'expression neutre est présente pour la majeure partie de la vidéo. Il s'avère que cette normalisation rend la détection d'AU bien plus performante, avec par exemple un triplement du F1-score pour l'AU 17 (chin rise) sur les datasets utilisés.

II.2.2) – vectorisation du texte

Le modèle utilisé pour vectoriser le texte est un BART (Bidirectional and Auto-Regressive Transformers), modèle proposé par Meta AI en 2019, et montrant de meilleures performances qu'un BERT simple. Le modèle repose essentiellement sur les transformers de Vaswani et al. (2017) [13], dont les variantes dominent dans la plupart des tâches NLP, comme cela peut se voir sur des benchmarks du type GLUE [21]. Ce modèle généralise BERT et GPT, en reprenant des parties de leurs architectures respectives, ce qui lui permet d'être performant sur un nombre plus important de tâches et même d'y établir de nouveaux records, et surtout d'être plus flexible que ceux-ci, ce qui lui permet d'accomplir un plus grand nombre de tâches. Cela passe par le pré-entraînement de BART sur des tâches plus variées que les deux modèles dont il s'inspire : des tâches de remplaçage de tokens effacés, de détermination du début d'un document, de remise dans l'ordre de phrases, … J'utilise le checkpoint bart-large trouvable sur HuggingFace et pré-entraîné sur la langue anglaise.

II.2.3) – vectorisation du son

Le son est la modalité la plus complexe à featuriser. J'utilise pour cela le Wav2vec2.0 de Meta AI [8], qui procède de manière similaire à BART pour générer des vecteurs de features, en apprenant à compléter des vecteurs de tokens lacunaires.

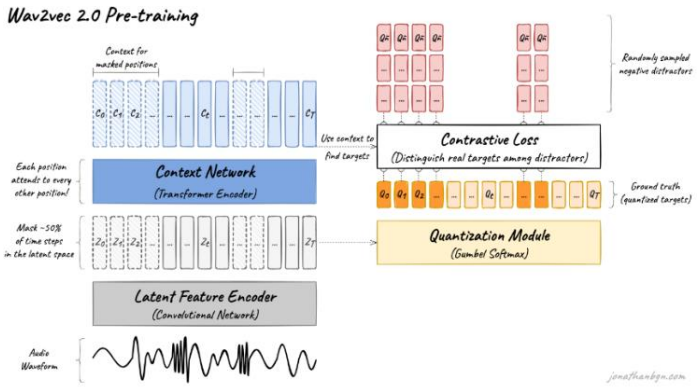


Figure 12 : Wav2vec2.0

En premier lieu, le modèle featurise le son par CNN temporels (« *Latent Feature Encoder* »). Cette première composante commence par normaliser le son RAW, et le fait passer par sept couches de CNN temporels avec une stride et une profondeur de noyau décroissantes, suivies d'une *Layer Normalization* et d'une GeLU.

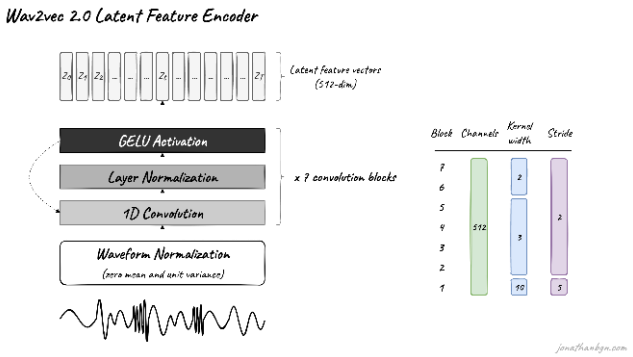


Figure 13 : les 7 couches de CNN temporels et leurs propriétés

L'output z de ce premier module, dont les composantes représentent chacune 25 ms de l'audio, est alors soumis à deux traitements en parallèle :

- On lui ajoute une sorte de positional encoding qui repose sur des CNN : $z' = z + GeLU(CNN(z))$. Z' est ensuite masqué à 50 %, c'est-à-dire que des composantes de Z' sont remplacées par un même vecteur obtenu par training. Z' est alors inséré dans

l'encoder d'un transformer, ce qui permet de produire une représentation qui incorpore des informations provenant de toute la séquence audio, sous la forme de vecteurs c_t ;

- Le son étant par nature continu, on le discrétise en *Speech Units*, unités plus petites que des phonèmes, par une *Product Quantization* [22]. Un z_t représente plusieurs de ces speech units : le but est de trouver automatiquement ces speech units dans des « codebooks », sortes de dictionnaires de toutes les speech units découvertes pendant le training, en choisissant ceux qui maximisent une certaine quantité. On utilise pour cela le Gumbel softmax [23], qui permet de rendre l'échantillonnage discret de quantizers différentiable, afin de rendre le module de quantization entraînable. On obtient alors une autre représentation du son avec les q_t (cf. figure 15)

Une fois les représentations c_t et q_t produites, le modèle s'entraîne grâce à une fonction de perte contrastive de la forme $L = L_m + \alpha L_d$, où L_m s'écrit :

$$L_m = -log \frac{exp(sim(c_t, q_t)/\kappa)}{\sum_{\tilde{q} \epsilon Q_t} exp(sim(c_t, \tilde{q})/\kappa)}$$

Figure 14 : composante L_m de la fonction de perte qui repose sur les similarités cosinus

Les c_t passent dans une couche linéaire afin d'être à la même dimension que les q_t. Ensuite, pour chaque position masquée t, le modèle va générer une centaine de fausses quantizations $q'_{t'}$ provenant d'autre positions dans la même phrase, et, grâce à la forme de la fonction de perte L_m, va encourager la plus forte similarité cosinus possible entre c_t et le vrai q_t, et pénaliser les fortes similarités entre c_t et les fausses représentations. La composante L_d (d pour « diversity ») encourage quant à elle l'utilisation de tout le codebook, elle est la fonction de perte du module de quantization.

La bande sonore du dialogue est alors vectorisée, sous la forme des vecteurs c_t.

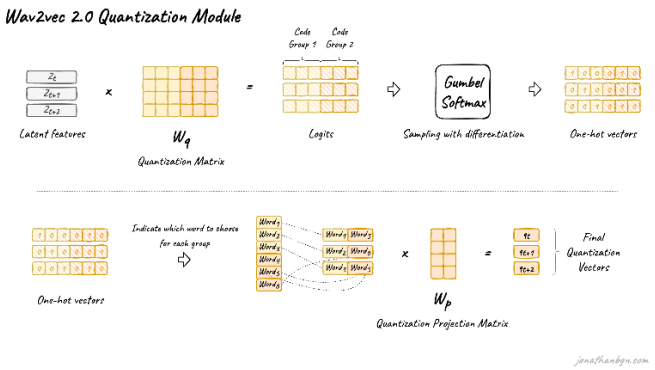


Figure 15 : quantization de la représentation du son

II.3) – L'architecture du modèle M2FNet

Le modèle M2FNet [12], pour « Multi-modal Fusion Network », repose largement sur les transformer encoders.

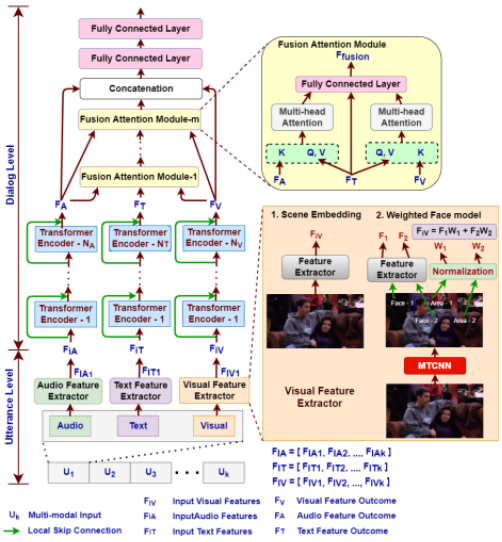


Figure 16 : architecture du modèle M2FNet (source : [12])

Les utterances, qui sont les inputs du modèle, commencent par être featurisées selon leurs trois modalités, comme cela a été vu dans la partie II.2). Puisqu'il y a plusieurs visages sur la vidéo, on utilise une moyenne pondérée (de coefficients appris) des features des visages, ce qui mène à de meilleures performances comme le montre le papier du modèle [12]. Ensuite, à l'aide d'un empilement de transformer encoders entourés de skip connections, on produit, pour chaque modalité séparément, une représentation qui incorpore des informations provenant de tout le dialogue. Cela a pour effet de produire des features encore plus représentatives du dialogue pour chaque modalité, en capturant la dynamique de la conversation. Après cela, les modalités sont fusionnées grâce à un module appelé *Fusion Attention Module*.

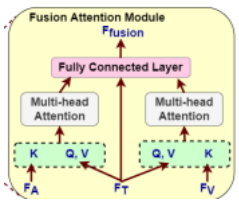


Figure 17 : Fusion Attention Module

Ceux-ci reposent sur les modules de multi-head attention. Dans le premier module Fusion Attention Module, on passe les features textuelles en Query et en Value de deux multi-head

attentions, et on passe en Key les features audio d'un côté, et les features visuelles de l'autre. Les sorties des deux multi-head attention sont concaténées, puis passées dans un réseau de neurones à une couche cachée afin de produire un premier vecteur F_{fusion} de features, qui tient compte à la fois de l'entièreté du dialogue, et de toutes les modalités. Sur les blocs suivants, c'est le vecteur F_{fusion} du bloc précédent qui est passé en Query et en Value, mais ce sont toujours les mêmes features audios et vidéos qui sont passées en Key. On incorpore ainsi progressivement au texte les informations sur l'émotion issues de l'audio et de la vidéo. Le vecteur F_{fusion} du dernier bloc est alors concaténé avec les vecteurs de features audios et vidéos, et cette concaténation est passée dans un réseau de neurones à deux couches cachées, pour produire une décision sur l'émotion prédominante dans l'utterance passée en entrée.

Les performances de ce modèle reposent très fortement sur le texte : les résultats décrits dans le papier mettent en exergue ce fait, avec une ablation study démontrant que le texte seul permet d'atteindre une accuracy de 67.24 %, soit 0.6 % de moins à peine que le modèle multimodal. Mais cette différence met aussi en évidence le fait qu'audio et vidéo ont un intérêt certain dans la prédiction de l'émotion, et donc que le deep learning multimodal est un domaine qu'il convient de continuer à explorer.

III) Implémentation, entraînement et performances du modèle

III.1) L'implémentation de M2FNet

Après avoir effectué une revue de littérature en détection d'émotions dans les conversations, je me suis attelé à l'implémentation d'un modèle inspiré de M2FNet en langage Python, en utilisant le framework PyTorch.

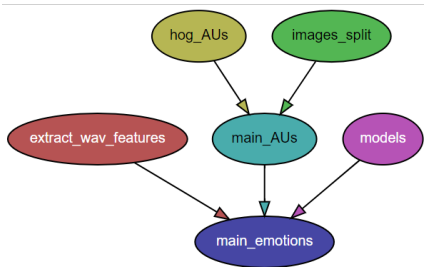


Figure 18 : dépendances entre les différentes composantes de "main_emotions", fonction d'entraînement du modèle

Le modèle est entraîné par main_emotions.py, laquelle fait appel à extract_wav_features.py pour featuriser le son, main_AUs.py pour featuriser les AUs (qui récupère les images de la

vidéo avec images_split.py et calcule les HOGs avec hog_AUs.py), et models.py qui featurise le texte et contient le modèle en PyTorch.

Un dernier fichier Python, non décrit dans le graphe de dépendance ci-dessus, permet de faire passer la vidéo d'un dialogue dans le modèle et de le labeliser avec l'émotion prédominante prédite.

L'architecture exacte de mon modèle est la suivante, dans l'ordre de passage :

- Un extracteur de features par modalité : on extrait ainsi un vecteur de taille 403 456 pour le texte, 32 768 pour les visages et 4 096 000 pour le son (ces deux derniers vecteurs étant complétés de zéros pour que les vecteurs issus de chaque utterance soient de tailles égales) ;
- Des couches fully connected qui réduisent les tailles des vecteurs précédents à 512 ;
- 6 transformer encoders empilés, d'entrées de taille 512, avec des skip connections entre eux, et avec 8 multi-head attentions chacun ;
- 10 Fusion Attention Modules, à 2*8 multi-head attention chacun, dont la sortie est de taille 512*3 ;
- Un réseau fully connected à deux couches cachées, et un softmax afin de réaliser une classification.

Les paramètres des réseaux de neurones sont initialisés par la méthode Glorot [24]

Des GeLUs sont utilisées entre chaque module, variante de la ReLU qui permet d'atteindre de meilleures performances.

III.2) L'entraînement du modèle

J'ai ensuite entraîné le modèle précédent sur le dataset MELD. Puisque celui-ci consiste en plusieurs milliers d'extraits vidéo, il m'a fallu faire usage du mésocentre de CentraleSupélec pour obtenir des temps de featurization et d'entraînement raisonnables. Ce mésocentre est celui des élèves, et propose des specs intéressantes :

- CPU : Intel Xeon® W-2225 @ 4.1 GHz 32 GB ;
- GPU : Geforce 3090 24 GB RAM) dont l'allocation se fait par des requêtes Slurm, au format shell script, qui précisent les capacités demandées, le gpu à utiliser, les capacités demandées...

Malgré la puissance du mésocentre, l'extraction de features sur les 13 000 utterances a duré une vingtaine d'heure, et l'entraînement du modèle en a duré une.

L'entraînement s'effectue avec un taux d'apprentissage constant de $3*10^{\wedge-4}$, avec des batches de taille 32 pendant 5 epochs. L'ensemble de training représente 60 % du dataset, l'ensemble de validation en représente 15 %, et l'ensemble de test en contient 25 %. La fonction de perte est une Cross Entropy Loss pondérée de façon à tenir compte des importantes différences entre les nombres d'occurrences de chaque émotion dans le dataset. La descente de gradient est effectuée à l'aide de l'optimizer Adam.

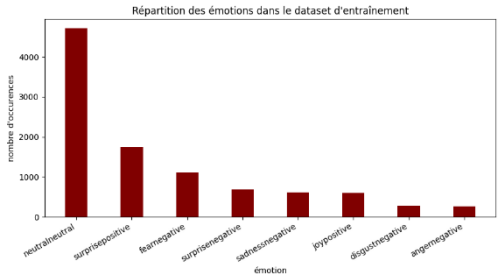


Figure 19 : répartition des émotions dans le dataset d'entraînement

III.3) Résultats

Les résultats finaux sont assez décevants, au regard de l'apparente solidité des extracteurs de features de mon modèle et des performances du papier dont je m'inspire. Ceux-ci sont exposés dans le tableau suivant, qui reprend deux des tableaux de résultats du papier M2FNet et qui les trie selon leur F1-score pondéré :

Modèle	**Accuracy (en pourcentage)**	**F1-score pondéré**
M2FNet (2022)	67.85 %	66.71
M2FNet, texte seulement	67.24 %	66.23
Xie et al. (Transformers, 2021)	65 %	64
DialogueTRM (Transformers, 2021)	65.66 %	63.55
DialogRNN (Recurrent NN, 2018)	59.54 %	57.03

BC-LSTM-Att (LSTM et Attention, 2017)	57.20 %	56.44
M2FNet, audio seulement	49.04 %	39.63
M2FNet, vidéo seulement	45.63 %	32.44
Mon modèle	35 %	

On considère l'accuracy classique, c'est-à-dire le pourcentage de prédictions correctes effectuées sur l'ensemble de test. Par manque de temps, je n'ai pas pu calculer pour mon modèle le F1-score pondéré, deuxième métrique pertinente considérée dans le papier de M2FNet (ce score est pondéré pour tenir compte de la disparité dans l'occurrence des différents labels dans le dataset).

L'accuracy maximale que j'ai obtenu sur l'ensemble de test est de 35 %, assez loin de l'accuracy de 68 % atteinte par M2FNet. Cela s'explique sans doute par trois choses :

- La période durant laquelle j'ai pu développer ce modèle étant assez faible, il y a d'une part beaucoup de pistes d'amélioration du training que je n'ai pas prises en compte et qui auraient grandement amélioré le modèle, comme la recherche d'optimizers plus performants qu'Adam, de taux d'apprentissage non-constants, d'hyperparamètres optimaux par grid search, … ;
- Il y a d'autre part quelques approximations que j'ai effectuées qui ont probablement beaucoup diminué l'accuracy. J'ai en effet dû compresser certains vecteurs de features, en les réduisant localement à leur moyenne (c'est-à-dire en remplaçant des blocs de features consécutives par leur moyenne) à cause des limites techniques du mésocentre de calcul. Au début de ce stage, je disposais d'un autre mésocentre de calcul capable de travailler avec les vecteurs entiers, mais celui-ci était en panne durant le dernier mois de mon stage ;
- Il y a également le problème de la reproductibilité du modèle M2FNet, que ses auteurs n'ont pas publié sur GitHub, et dont le papier est souvent trop sommaire sur les détails de l'implémentation. A cause de la contrainte de temps et de la difficulté à reproduire le papier, j'ai dû simplifier le modèle, en implémentant des extracteurs de features figés et pré-entraînés plutôt que des extracteurs de features qui s'affinent au fur et à mesure de l'entraînement, et une loss function consistant en une simple Cross Entropy Loss pondérée, plutôt que celle proposée dans le papier qui fait intervenir une Triplet

Loss sur les extracteurs de features pour les renforcer, en plus de la Cross Entropy Loss.

Il y a également quelques pistes, non explorées dans le papier, qui auraient pu être mises en œuvre, comme effectuer une concaténation de tous les vecteurs de features des visages plutôt qu'une moyenne pondérée de ceux-ci, ce qui aurait peut-être conduit à des vecteurs de features plus représentatifs. Puisque c'est le texte qui explique principalement les performances des meilleurs modèles, j'aurais pu considérer également un modèle proche de celui-ci, mais avec une fusion de modalités moins « forte », qui laisse plus de place au texte, comme un système de pondération des modalités par exemple.

Conclusion

Même si les résultats obtenus dans la pratique sont assez décevants au regard de ce qui est atteint dans l'état de l'art, ces trois mois de travaux de revue de littérature et de tentative d'implémentation auront un intérêt pour mes encadrantes, qui ont l'intention de reproposer le sujet à des élèves, voire d'en faire un sujet de thèse.

Ce stage m'a également été très utile sur le plan personnel : j'ai pu me former au framework PyTorch, étudier dans le détail des papiers incontournables dans le milieu du deep learning, et plus précisément dans le milieu du deep learning multimodal.

Bibliographie

[1] T. Baltrušaitis, M. Mahmoud and P. Robinson, "Cross-dataset learning and person-specific normalisation for automatic Action Unit detection", 2015

[2] G. Miriam Jacob and B. Stenger, "Facial Action Unit Detection With Transformers," 2021

[3] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space.

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation

[5] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding

[6] Lin, Y., Meng, Y., Sun, X., Han, Q., Kuang, K., Li, J., & Wu, F. (2021). Bertgcn: Transductive text classification by combining gcn and bert

[7] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding.

[8] Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations.

[9] Hsu, W. N., Bolte, B., Tsai, Y. H. H., Lakhotia, K., Salakhutdinov, R., & Mohamed, A. (2021). Hubert: Self-supervised speech representation learning by masked prediction of hidden units.

[10] Huang, SC., Pareek, A., Seyyedi, S. et al. Fusion of medical imaging and electronic health records using deep learning: a systematic review and implementation guidelines

[11] Li, Z., Tang, F., Zhao, M., & Zhu, Y. (2022). EmoCaps: Emotion Capsule based Model for Conversational Emotion Recognition.

[12] Chudasama, V., Kar, P., Gudmalwar, A., Shah, N., Wasnik, P., & Onoe, N. (2022). M2FNet: Multi-modal Fusion Network for Emotion Recognition in Conversation.

[13] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need.

[14] Shazeer, N. (2020). Glu variants improve transformer.

[15] Joshi, A., Bhat, A., Jain, A., Singh, A. V., & Modi, A. (2022). COGMEN: COntextualized GNN based Multimodal Emotion recognitioN.

[16] Joshi, R., & Gupta, A. (2022). Performance Comparison of Simple Transformer and Res-CNN-BiLSTM for Cyberbullying Classification.

[17] A. Zeyer, P. Bahar, K. Irie, R. Schlüter and H. Ney, "A Comparison of Transformer and LSTM Encoder Decoder Models for ASR," 2019

[18] Rosenberg Ekman. What the face reveals : Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)

[19] Baltrusaitis, Tadas & Robinson, Peter & Morency, Louis-Philippe. (2013). Constrained Local Neural Fields for Robust Facial Landmark Detection in the Wild

[20] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005*

[21] https://gluebenchmark.com/leaderboard

[22] H. Jégou, M. Douze and C. Schmid, "Product Quantization for Nearest Neighbor Search," (2010)

[23] Jang, E., Gue S., & Poole, B. (2016). Categorical reparameterization with gumbel-softmax.

[24] Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks.

[25] Deng, J., Guo, J., Ververas, E., Kotsia, I., & Zafeiriou, S. (2020). Retinaface: Single-shot multi-level face localisation in the wild.

[26] Clark, E., Kessinger, J., Duncan, S., Bell, M., Lahne, J., Gallagher, D., O'Keefe, S. (2020. The Facial Action Coding System for Characterization of Human Affective Response to Consumer Product-Based Stimuli: A Systematic Review

[27] https://en.wikipedia.org/wiki/Mel-frequency_cepstrum (consultée en octobre 2022)