Un modèle unifié pour la classification de signaux sur graphe avec de l'apprentissage profond

Myriam BONTONOU[1,2], Carlos LASSANCE[1,2], Jean-Charles VIALATTE[1], Vincent GRIPON[1,2]

[1]Lab-STICC, Département électronique
IMT Atlantique, Brest, France

[2]MILA, Département d'informatique et de recherche opérationnelle
Université de Montréal, Montréal, Canada

myriam.bontonou@imt-atlantique.fr

Résumé – Nous proposons de passer en revue quelques-uns des principaux modèles utilisés pour la classification supervisée de signaux sur graphe avec de l'apprentissage profond. L'objectif de la classification supervisée de signaux sur graphe est de classer un signal dont les composantes sont définies sur les sommets d'un graphe. Le réseau de neurones convolutif (CNN) est très performant pour classer des signaux définis sur un graphe « grille » (comme les images). Cependant, comme il ne peut pas être utilisé sur des signaux définis sur un graphe quelconque, d'autres modèles sont apparus, essayant d'appliquer ses propriétés à n'importe quel graphe. L'objectif général de cette étude est de comparer certains des principaux modèles de classification supervisée des signaux sur graphe. Nous proposons également un formalisme unifié.

Abstract – We propose to review some of the major models performing supervised classification of graph signals in deep learning. The goal of supervised classification of graph signals is to classify a signal whose components are defined on the vertices of a graph. Convolutional Neural Networks (CNN) are very efficient at classifying signals defined on a grid graph (such as images). However, as they can not be used on signals defined on an arbitrary graph, other models have emerged, aiming to extend its properties to any graph. The overall purpose of this study is to provide a comparison of some of the major models in supervised classification of graph signals. We also introduce a unified formalism.

1 Introduction

Les réseaux d'apprentissage profond atteignent les meilleures performances pour de nombreux problèmes de l'apprentissage machine. Ces modèles s'appuient sur des réseaux de neurones, organisés comme un assemblage de couches. Chaque couche compose une fonction linéaire, dont les coefficients sont les paramètres à apprendre, et une fonction non linéaire. Le réseau est entraîné de bout-en-bout, en utilisant une variante de la descente de gradient sur un critère à optimiser. Un des facteurs expliquant les performances de ces réseaux est le filtrage par convolution, utilisé dans de nombreux domaines comme la vision, et dont le principe est de calculer le produit de convolution entre des portions de l'entrée et un filtre. Les coefficients des filtres deviennent les paramètres entraînables du réseau.

Deux caractéristiques distinguent ces filtres d'une fonction linéaire quelconque : a- les mêmes paramètres sont utilisés sur toutes les portions de l'entrée, b- les sorties du réseau dépendent seulement d'une portion locale de l'entrée. De ces deux caractéristiques découle une propriété essentielle des couches convolutives : le nombre de paramètres devient indépendant de la taille de l'entrée. Malheureusement elles ne sont définies que si leur entrée repose sur un espace Euclidien discret (comme les séquences audio ou les images). Or, beaucoup de données ne sont pas définies sur un tel espace mais disposent d'une topologie plus complexe, comme par exemple les nuages de points, les réseaux de capteurs géo-localisés, ou l'activité fonctionnelle dans le cerveau. Ainsi, ces dernières années, une variété de travaux a émergé pour étendre les capacités des couches de convolution à des données plus diverses, et notamment à des données définies sur des graphes.

Dans cette article, nous nous intéressons donc au problème de la classification supervisée de signaux sur graphes. Nous introduisons les principales méthodes proposées dans la littérature et montrons qu'elles peuvent s'exprimer sous un formalisme unifié, permettant de leur donner une lecture nouvelle et comparative. Nous réalisons ensuite des comparaisons de ces méthodes sur des jeux de données construits pour l'occasion.

2 Principales méthodes

Un graphe est un tuple $\mathcal{G} = \langle V, \mathbf{A} \rangle$, où $V = \{1, \dots, N\}$ est un ensemble fini de sommets, et $\mathbf{A} \in \mathbb{R}^{N \times N}$ est la matrice d'adjacence. Dans le cas d'un graphe non pondéré, $\mathbf{A}_{ij}$ est un indicateur binaire de l'existence d'un lien entre les sommets i et j. Dans le cas d'un graphe pondéré, $\mathbf{A}_{ij}$ correspond au poids du lien s'il existe et vaut 0 sinon. Dans certains cas, il est utile d'associer des caractéristiques aux sommets du graphe. L'ensemble de ces caractéristiques forme une matrice $\mathbf{X} \in \mathbb{R}^{N \times P}$

où $\mathbf{X}_{ip}$ représente la p^{ieme} caractéristique du sommet i.

Deux types d'approche ont été proposés pour analyser les signaux sur graphes avec des modèles d'apprentissage profond. Le premier s'inspire de la théorie spectrale des graphes, où la transformation de Fourier discrète (TFD) est étendue aux signaux sur graphe. Le deuxième type d'approche cherche à étendre la propriété de localité de la convolution à un graphe.

2.1 Modèles inspirés de la théorie spectrale

Considérons un graphe $\mathcal{G} = \langle V, \mathbf{A} \rangle$ et sa matrice laplacienne $\mathbf{L} = \mathbf{D} - \mathbf{A}$ ($\mathbf{D}$ est la matrice diagonale telle que $\mathbf{D}_{ii} = \sum_{j=1}^{N} A_{ij}$.) La matrice $\mathbf{L}$ étant symétrique réelle, elle admet un ensemble de vecteurs propres orthonormaux $\{\mathbf{u}_l\}_{l=1}^{N} \in \mathbb{R}^N$ et un ensemble de valeurs propres associées $\{\lambda_l\}_{l=1}^{N}$.
Soit $\mathbf{U} = [\mathbf{u}_1, ..., \mathbf{u}_N] \in \mathbb{R}^{N \times N}$, $\mathbf{U}^*$ la transposée de $\mathbf{U}$ et $\mathbf{\Lambda} = diag([\lambda_1, ..., \lambda_N]) \in \mathbb{R}^{N \times N}$, $\mathbf{L}$ se décompose en $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*$. Par analogie avec la TFD, la transformation de Fourier sur graphe (TFG) [1] de $\mathbf{X} \in \mathbb{R}^{N \times P}$ se définie par : $\hat{\mathbf{X}} = \mathbf{U}^*\mathbf{X}$ et la transformation inverse par : $\tilde{\mathbf{X}} = \mathbf{U}\ \hat{\mathbf{X}}$.
Par analogie avec le théorème de convolution, la convolution sur graphe [2] est définie comme le produit de Hadamard ($\odot$) dans le domaine spectral du graphe. Soit une caractéristique du signal d'entrée $\mathbf{x}_p = \mathbf{X}[:, p] \in \mathbb{R}^N$ et une caractéristique de sortie q. Le filtre reliant la caractéristique d'entrée p à la sortie q est représenté par un vecteur $\mathbf{w}_{pq} \in \mathbb{R}^N$, tel que :

$$\mathbf{x}_p \circledast \mathbf{w}_{pq} = \widetilde{\hat{\mathbf{x}}_p \odot \hat{\mathbf{w}}_{pq}} \ . \tag{1}$$

Cette formule a été transférée aux réseaux de neurones [3] pour définir un équivalent des couches de convolution classiques. Le vecteur des coefficients $\mathbf{w}_{pq}$ remplace le vecteur des coefficients de la fonction linéaire g de l'une des couches :

$$g(\mathbf{X})_q = \mathbf{U} \sum_{p=1}^{P} \mathbf{w}_{pq} \odot \mathbf{U}^*\mathbf{x}_p \ . \tag{2}$$

Cependant, ce modèle a deux défauts : a- le nombre de paramètres à apprendre dépend du nombre de sommets du graphe ($\Theta(N)$ paramètres), b- il nécessite de calculer la TFG (complexité en $\mathcal{O}(N^3)$).

ChebNet Un des moyens de contourner ces défauts est d'approximer la TFG avec des polynômes de Tchebychev [2]. Cette approximation a aussi été transférée aux réseaux de neurones [4]. Soit k l'ordre d'un polynôme. Notons les polynômes de Tchebychev T_k. La caractéristique de sortie q d'un signal sur graphe $\mathbf{X} \in \mathbb{R}^{N \times P}$ devient :

$$g(\mathbf{X})_q = \sum_{p=1}^{P} \mathbf{W}_{pq}(\mathbf{L})\mathbf{x}_p \ . \tag{3}$$

Soir une constante K, $\lambda_{\max}$ la plus grande des valeurs propres de $\mathbf{L}$, la matrice identité $\mathbf{I}_N$ de dimension N, $T_k(\frac{\lambda_{\max}\mathbf{L}}{2} - \mathbf{I}_N) \in \mathbb{R}^{N \times N}$ et $\theta \in \mathbb{R}^K$. La matrice des coefficients $\mathbf{W}_{pq} \in \mathbb{R}^{N \times N}$ est définie comme :

$$\mathbf{W}_{pq}(\mathbf{L}) = \sum_{k=0}^{K-1} \theta_k\ T_k(\frac{\lambda_{\max}\mathbf{L}}{2} - \mathbf{I}_N) \ . \tag{4}$$

Les $P \times Q$ vecteurs $\theta \in \mathbb{R}^K$ contiennent les paramètres du réseau. Ce nombre est indépendant du nombre de sommets du graphe. De plus, ce modèle retrouve une des propriétés de la convolution classique : les sorties du réseau dépendent seulement d'une portion locale de l'entrée. En effet, une fonction représentée par un polynôme de la matrice laplacienne d'ordre k est exactement k-localisée – la valeur de sortie attribuée à un sommet dépend seulement des valeurs des sommets qui sont au plus à k connections de lui.

Des variantes de cette méthode existent : base canonique spectrale fixée [5], utilisation d'ondelettes [6, 7] ou d'autres types de polynômes [8]. Les modèles inspirés de la théorie spectrale ont cependant deux défauts : a- ils ne sont pas robustes aux modifications de la structure du graphe (un graphe légèrement différent génère un signal dans le domaine spectral $\hat{\mathbf{X}}$ parfois très différent), b- ils sont isotropes, ce qui ici signifie que la valeur de chaque sommet dépend identiquement des valeurs de tous les sommets à k-connections de lui, ce qui pose problème pour certaines applications.

2.2 Modèles inspirés de la diffusion

Dans une autre famille de méthodes, la convolution sur graphe a été directement pensée comme une fonction linéaire localisée. L'entrée de la fonction linéaire g est le signal $\mathbf{X} \in \mathbb{R}^{N \times P}$. g produit un nouveau signal $g(\mathbf{X}) \in \mathbb{R}^{N \times Q}$ avec un nombre de caractéristiques q potentiellement différent de p. Soit un sommet j, $g(\mathbf{X})_j$ est une fonction linéaire des valeurs des sommets reliés à j sur le graphe. La difficulté consiste à utiliser le même nombre de paramètres quelque soit le nombre de sommets dans le graphe. De nombreux articles s'attaquent à ce problème mais, au final, leurs performances sont très similaires. Dans cette revue, nous avons sélectionné trois méthodes couramment utilisées.

« Graph Convolutional Network » GCN est un réseau de neurones utilisant la fonction linéaire g suivante [9] :

$$g(\mathbf{X}) = \tilde{\mathbf{A}}\ \mathbf{X}\ \mathbf{\Theta} \ , \tag{5}$$

avec $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}$, $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$, $\hat{\mathbf{D}}$ matrice des degrés de $\hat{\mathbf{A}}$, $\mathbf{\Theta} \in \mathbb{R}^{P \times Q}$. $\tilde{\mathbf{A}}\mathbf{X}$ diffuse le signal sur le graphe. Les valeurs du signal associé à un sommet dépendent maintenant des valeurs des signaux des sommets voisins. $\mathbf{\Theta}$ apprend plusieurs représentations du signal diffusé. Ce modèle simple se rapproche en réalité de ChebNet si on le reformule ainsi :

$$g(\mathbf{X}) = \sum_{c=0}^{C-1} T_c\left(\frac{\lambda_{\max}\mathbf{L}}{2} - \mathbf{I}_N\right)\ \mathbf{X}\ \mathbf{\Theta} \ . \tag{6}$$

« Graph ATtention network » GAT [10] apprend l'importance à accorder aux voisins d'un sommet. Étant donné une matrice $\mathbf{\Theta} \in \mathbb{R}^{P \times Q}$ et un couple de sommets voisins (i, j), une couche entièrement connectée par un vecteur $\mathbf{a} \in \mathbb{R}^{2Q}$ apprend l'attention α_{ji} que i donne à j. $\alpha_{ji} = softmax(e_{ji})$, avec $e_{ji} = LeakyReLU(\mathbf{a}^T\ [\mathbf{X}_j\mathbf{\Theta}\ ||\ \mathbf{X}_i\mathbf{\Theta}])$, $||$ signifiant

la concaténation. $\mathbf{\Theta}$ transforme les caractéristiques d'entrée en caractéristiques de plus haut niveau, et $\mathbf{a}$ diffuse le signal. Pour exploiter la structure du graphe, un mécanisme appelé masque d'attention est mis en place : e_{ji} vaut zéro si le sommet i n'est pas connecté au sommet j. Finalement, en notant $\mathcal{R}(j)$ l'ensemble des sommets connectés au sommet j, on obtient la fonction linéaire g suivante :

$$g(\mathbf{X})_j = \sum_{i \in \mathcal{R}(j)} \alpha_{ji} \mathbf{X}_i \mathbf{\Theta} \ . \tag{7}$$

Pour stabiliser l'apprentissage, plusieurs fonctions g sont calculées indépendamment, puis concaténées ou moyennées.

« Topology Adaptative GCN » Un autre modèle très similaire à GCN fait en sorte que le signal sur un sommet dépendent des signaux de ses voisins situés à au plus C connections. Ce modèle appelé « Topology Adaptative GCN » (TAGCN) [11] introduit les puissances de la matrice d'adjacence normalisée $\tilde{\mathbf{A}}$ dans le modèle GCN :

$$g(\mathbf{X}) = \sum_{c=1}^{C} \tilde{\mathbf{A}}^{\,c} \ \mathbf{X} \ \mathbf{\Theta}_c \ . \tag{8}$$

3 Formalisme unifié

Comme chaque auteur introduit son propre formalisme, il peut être ardu de comparer les modèles. Nous proposons un unique formalisme qui permet de représenter tous les modèles existants. Une couche d'un réseau de neurones compose une fonction linéaire g avec une fonction non linéaire. Les modèles proposent de nouvelles fonctions linéaires. Ce sont ces dernières que nous formalisons. Les indices sont notés en minuscule, et la taille des ensembles d'indices en majuscule.

En apprentissage profond, les données d'entraînement sont divisées en B lots. L'entrée d'une couche est un tenseur $\mathbf{X} \in \mathbb{R}^{\mathrm{B} \times \mathrm{P} \times \mathrm{I}}$. i représente un neurone en entrée de la couche. p représente un canal. Par exemple, dans le cas d'un ensemble de données constitué d'images en couleur, en entrée de la première couche du réseau, un neurone correspond à un pixel de l'image ; un canal correspond à une des trois couleurs primaires. De façon similaire, $g(\mathbf{X}) \in \mathbb{R}^{\mathrm{B} \times \mathrm{Q} \times \mathrm{J}}$. j représente un neurone en sortie de la couche. q représente une caractéristique (indépendante des autres caractéristiques).

Pour expliquer notre formalisme, introduisons un premier exemple simple. Imaginons que nous mettons toutes les images d'entraînement dans un seul lot. L'entrée d'une couche se réduit alors à un tenseur $\mathbf{X} \in \mathbb{R}^{\mathrm{P} \times \mathrm{I}}$. Nous stockons les coefficients de la fonction linéaire g dans un tenseur $\mathbf{W} \in \mathbb{R}^{\mathrm{Q} \times \mathrm{P} \times \mathrm{J} \times \mathrm{I}}$. Pour chaque caractéristique q et pour chaque neurone de sortie j, nous obtenons la formule suivante :

$$g(\mathbf{X})_{qj} = \sum_{p=1}^{\mathrm{P}} \sum_{i=1}^{\mathrm{I}} \mathbf{W}_{qpji} \mathbf{X}_{pi} \ .$$

Parfois, les mêmes coefficients sont utilisés pour plusieurs connections. Il est donc peu efficace de les stocker tous dans le tenseur $\mathbf{W}$. Par la suite, nous appelons « paramètres » les coefficients uniques. Nous proposons de stocker tous les paramètres dans un vecteur $\theta \in \mathbb{R}^{\mathrm{K}}$. K correspond au nombre de paramètres. Nous définissons un tenseur d'allocation $\mathbf{S} \in \mathbb{R}^{\mathrm{K} \times \mathrm{J} \times \mathrm{I}}$ tel que : $\mathbf{W}_{qpji} = \sum_{k=1}^{\mathrm{K}} \theta_{qpk} \mathbf{S}_{kji}$. Pour simplifier les notations, nous omettons d'écrire les sommes et par convention, nous considérons qu'il y a une somme sur un indice dès lors qu'il n'est plus présent dans la variable de sortie. La formule ci-dessus devient : $\mathbf{W}_{qpji} = \theta_{qpk} \mathbf{S}_{kji}$. En réintroduisant les indices des lots, nous obtenons le formalisme suivant :

$$g(\mathbf{X}) = \widehat{\mathbf{\Theta}\mathbf{S}\mathbf{X}} \text{ avec } \left\{ \begin{array}{c} \mathbf{W}_{qpji} = \mathbf{\Theta}_{qpk} \mathbf{S}_{kji} \\ g(\mathbf{X})_{bqj} = \mathbf{W}_{qpji} \mathbf{X}_{bpi} \end{array} \right\} \tag{9}$$

La représentation ternaire $g(\mathbf{X}) = \widehat{\mathbf{\Theta}\mathbf{S}\mathbf{X}}$ distingue les rôles de $\mathbf{\Theta}$ et de $\mathbf{S}$. Nous pouvons maintenant réécrire les quatre modèles comparés dans la section 4 ainsi qu'un réseau de neurones classique et qu'un réseau de neurones convolutif classique.

Ces modèles ont une précision similaire sur les données de référence actuellement utilisées dans la communauté. Cela peut s'interpréter de deux façons : soit les modèles sont trop similaires, soit les données de référence ne permettent pas de différencier leurs performances. Nous introduisons dans la prochaine partie deux protocoles de test permettant de mieux différencier ces modèles.

4 Résultats

Pour commencer, nous utilisons un ensemble de données créé à partir de petites images pour vérifier si les méthodes sont capables de reproduire la performance des réseaux convolutifs sur une tâche qui les favorise (données structurées de façon régulière). Nous avons choisi la base de données CIFAR10 ([12]), qui contient des images en couleurs de 32x32 pixels réparties dans 10 catégories. Nous utilisons un réseau simple à 4 couches suivi d'une couche de regroupement et d'une dernière couche sans structure (entièrement connectée) pour la classification. Malheureusement, nous ne sommes pas capables de tester GAT avec l'architecture choisie, car le modèle dépasse la mémoire de notre carte graphique. Les résultats sont décrits dans la table 1. ChebNet se distingue des autres modèles avec de bien meilleures performances, ce qui n'est pas trop surprenant compte tenu des résultats présentés dans [13]. Une étude approfondie de la structure du réseau et des hyperparamètres serait nécessaire pour s'assurer que les autres modèles ne parviennent vraiment pas à approcher cette précision. Nous tirons deux conclusions de ce premier test : a- aucun des modèles considérés ne parvient réellement à approcher les performances du réseau convolutif de référence, impliquant que les modèles n'exploitent pas totalement la structure sous-jacente (de meilleures performances peuvent être atteintes en ne s'appuyant que sur le graphe [13]), b- de tous les modèles comparés, ChebNet est le plus performant sur cette tâche s'exécutant sur un graphe peu irrégulier.

Nous nous intéressons ensuite à une tâche pour laquelle un réseau convolutif ne peut être utilisé directement. Nous utili-

TABLE 1 – Performance des méthodes sur la base de donnée CIFAR-10.

Structure	Modèle	Taux d'erreur
–	CNN	16.92%
Graphe grille	ChebNet	28.1%
	GCN	51.96%
	GAT	Pas assez de mémoire
	TAGCN	51.74%
Graphe inferé	ChebNet	29.46%
	GCN	50.33%
	GAT	Pas assez de mémoire
	TAGCN	51.67%

sons ici une base de donnée EEG [14]. Le graphe est un graphe de voisinage sur la structure 3D irrégulière et nous comparons avec les résultats obtenus en faisant une projection azimutale pour obtenir une structure 2D régulière. Nous utilisons l'architecture proposée dans [14], avec 2 couches convolutives suivies de 2 couches linéaires. Les résultats sont décrits dans la table 2. Dans ce cas, comme la structure est plus adaptée aux graphes qu'à la grille 2D régulière, les méthodes sur graphes permettent de dépasser les performances du réseau convolutif de référence.

TABLE 2 – Performance des méthodes sur les données EEG.

Structure	Modèle	Taux d'erreur
–	MLP équivalent	12.95%
Projection azimutale	CNN équivalent [14]	13.05%
	CNN plus grand [14]	12.39%
Graphe de voisinage	ChebNet	10.22%
	GCN	11.37%
	GAT	16.75%
	TAGCN	10.54%

5 Conclusion et perspectives

Nous avons introduit un formalisme unifié pour les couches traitant des signaux sur graphes dans le cadre de l'apprentissage profond. Nous avons introduit deux bases de données afin de comparer ces méthodes dans des tâches d'apprentissage supervisé de signaux sur graphes. Notre étude montre que ces méthodes ne parviennent pas à totalement exploiter la structure sous-jacente, mais parviennent néanmoins à améliorer les performances par rapport à des techniques simplifiant la structure des signaux, lorsque celle-ci est très irrégulière.

Références

[1] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs : Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3) :83–98, 2013.

[2] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2) :129–150, 2011.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv :1312.6203*, 2013.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[5] Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. Syncspeccnn : Synchronized spectral cnn for 3d shape segmentation. In *CVPR*, pages 6584–6592, 2017.

[6] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1872–1886, 2013.

[7] Xu Chen, Xiuyuan Cheng, and Stéphane Mallat. Unsupervised deep haar scattering on graphs. In *Advances in Neural Information Processing Systems*, pages 1709–1717, 2014.

[8] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets : Graph convolutional neural networks with complex rational spectral filters. *arXiv preprint arXiv :1705.07664*, 2017.

[9] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv :1609.02907*, 2016.

[10] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv :1710.10903*, 2017.

[11] Jian Du, Shanghang Zhang, Guanhang Wu, José MF Moura, and Soummya Kar. Topology adaptive graph convolutional networks. *arXiv preprint arXiv :1710.10370*, 2017.

[12] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[13] Carlos Eduardo Rosar Kos Lassance, Jean-Charles Vialatte, and Vincent Gripon. Matching convolutional neural networks without priors about data. In *2018 IEEE Data Science Workshop (DSW)*, pages 234–238. IEEE, 2018.

[14] Pouya Bashivan, Irina Rish, Mohammed Yeasin, and Noel Codella. Learning representations from eeg with deep recurrent-convolutional neural networks. *International Conference on Learning Representations*, 2016.