La CTC et son intrigant label "*blank*"

Étude comparative de méthodes d'entraînement de réseaux de neurones pour la reconnaissance d'écriture

Théodore Bluche [1] — Christopher Kermorvant [2] — Hermann Ney [3] — Jérôme Louradour [1]

[1] *A2iA SA, Paris France*
[2] *Teklia SASU, Paris, France*
[3] *RWTH, Human Language Technology and Pattern Recognition, Aachen, Allemagne*

RÉSUMÉ. *Les systèmes de reconnaissance d'écriture vainqueurs d'évaluations internationales ces dernières années sont basés sur des réseaux de type LSTM, entraînés avec un critère de classification temporelle connexionniste (CTC). L'algorithme de la CTC est basé sur une procédure "forward-backward", sans segmentation de la séquence d'entrée avant l'entraînement. Les sorties du réseau sont les caractères à modéliser, auxquels on ajoute un label spécial, non-caractère, (blank). D'autre part, dans les systèmes hybrides réseaux de neurones / modèles de Markov cachés (MMCs), les réseaux sont entraînés au niveau trame à prédire des états de MMC. Dans cet article, nous montrons que la CTC est une forme dérivée de l'entraînement forward-backward de MMCs, et qu'elle peut donc être étendue à des topologies arbitraires de MMC. Nous appliquons cette méthode à des perceptrons multicouches et la comparons à l'entraînement au niveau trame dans diverses situations. Enfin, nous analysons le rôle intrigant de ce label spécial "blank".*

ABSTRACT. *In recent years, Long Short-Term Memory Recurrent Neural Networks (LSTM-RNNs) trained with the Connectionist Temporal Classification (CTC) objective won many international handwriting recognition evaluations. The CTC algorithm is based on a forward-backward procedure, avoiding the need of a segmentation of the input before training. The network outputs are characters labels, and a special non-character label. On the other hand, in the hybrid Neural Network / Hidden Markov Models (NN/HMM) framework, networks are trained with framewise criteria to predict state labels. In this paper, we show that CTC training is close to forward-backward training of NN/HMMs, and can be extended to more standard HMM topologies. We apply this method to Multi-Layer Perceptrons (MLPs), and investigate the properties of CTC, especially the role of the special label.*

MOTS-CLÉS : *CTC, réseaux de neurones, reconnaissance d'écriture.*

KEYWORDS: *connectionist temporal classification, neural networks, handwriting recognition.*

1. Introduction

La classification temporelle connexionniste (CTC, (Graves *et al.*, 2006)) est une méthode pour entraîner des réseaux de neurones récurrents à étiqueter des séquences sans segmentation explicite. Cette méthode donne de très bons résultats avec des réseaux de type LSTM pour la reconnaissance d'écriture manuscrite, et ces modèles ont remporté de nombreuses évaluations internationales récentes (*e.g.* OpenHaRT (Tong *et al.*, 2014), Maurdor (Brunessaux *et al.*, 2014)).

Cette approche sans segmentation se rapproche des procédures d'entraînement des modèles de Markov cachés (MMCs) (*e.g.* l'algorithme de Baum-Welch (Rabiner et Juang, 1986)), qui ont été utilisées dans les années 90 pour entraîner des modèles hybrides réseaux de neurones (NN) / MMC, par exemple dans (Senior et Robinson, 1996 ; Hennebert *et al.*, 1997 ; Yan *et al.*, 1997). La principale différence avec ces méthodes est la contrainte imposée par la CTC au réseau d'avoir une sortie par caractère, plus une sortie spéciale "*non-caractère*" que nous appellerons "*blank*".

L'alternative la plus courante ces dernières années pour l'entraînement de systèmes hybrides NN/MMC, notamment pour les perceptrons multicouches (MLPs) consiste à entraîner d'abord un MMC pour récupérer les alignements forcés avec les données, permettant d'associer chaque trame à un état de MMC pour entraîner le réseau en classification de trames.

Dans cet article, nous étudions l'entraînement CTC, et nous nous efforçons de répondre aux questions suivantes :

– Mises à part les sorties du réseaux, une par caractère et une consacrée au label spécial *blank*, *quelles sont les différences entre l'entraînement CTC et les méthodes sans segmentation explicite proposées dans les années 90 ?*

– *Quelle est l'importance du nombre d'états de MMC dans les systèmes hybrides basés sur des MLPs et des RNNs ?*

– *Peut-on utiliser l'entraînement CTC avec des MLPs ? Améliore-t-il les résultats ?*

– *Comment l'entraînement CTC se compare-t-il à l'entraînement au niveau trame ?*

– *En quoi le label spécial est-il bénéfique pour la reconnaissance ? Quel est son rôle, et dans quelles conditions est-il bénéfique ?*

Nous articulons cette étude en trois grands axes. Dans un premier temps, nous menons une étude théorique pour comparer la CTC aux méthodes d'entraînement de systèmes hybrides NN/MMC, et nous montrons qu'elle en est en fait un cas particulier simplifié. Fort de ces observations, nous proposons dans un second temps une comparaison expérimentale de méthodes d'entraînement au niveau trame et de la CTC et ses différents aspects. En particulier nous étudions l'influence du nombre d'états de MMC par caractère (que nous appellerons par abus de langage *topologie* dans la suite de l'article) et l'impact du label *blank*, appliqué à des MLPs et des RNNs. Les ré-

sultats nous montrent une interaction particulière entre la CTC, ce label et les RNNs. Finalement, nous analysons le comportement de la CTC pour tenter d'apporter une explication au phénomène observé, et pour mieux comprendre le mécanisme de ce type d'apprentissage, ce qui nous permet par la suite de proposer des méthodes pour rendre l'entraînement plus efficace.

2. Relations entre la CTC et l'entraînement forward-backward des modèles hybrides NN/MMC

2.1. *Notations*

Soit $\mathcal{Q}_n(\mathbf{W})$ l'ensemble de toutes les séquences d'états de taille n représentant une séquence donnée de mots $\mathbf{W}$. Appelons $\mathcal{Q}(\mathbf{W})$ la liste de tous les états du modèle de $\mathbf{W}$. Cette liste peut contenir des doublons (le même caractère peut apparaître deux fois). $\mathcal{Q}^*$ est l'ensemble des identifiants d'états distincts. Enfin, nous appelons $\mu : \mathcal{Q}(\mathbf{W}) \mapsto \mathcal{Q}^*$ la fonction qui associe un identifiant à un état, ce qui sera nécessaire car les MMCs ont un seul modèle d'émission (représenté ici par l'identifiant) pour chaque élément de $\mathcal{Q}^*$, alors que dans l'algorithme de forward-backward les éléments de $\mathcal{Q}(\mathbf{W})$ sont considérés.

2.2. *Les équations de l'entraînement forward-backward des modèles hybrides NN/MMC*

L'entraînement forward-backward des réseaux de neurones se base sur l'utilisation des probabilités a posteriori fournies par le réseau dans la formulation pour les MMCs :

$$p(\mathbf{x}|\mathbf{W}) = \left(\prod_{t=1}^{|\mathbf{x}|} p(x_t)\right) \sum_{\mathbf{q}\in\mathcal{Q}_{|\mathbf{x}|}(\mathbf{W})} p(q_1,\mathbf{W}) \frac{p(q_1|x_1)}{p(q_1)} \prod_{t=2}^{|\mathbf{x}|} \frac{p(q_t|x_t)}{p(q_t)} p(q_t|q_{t-1},\mathbf{W}) \quad [1]$$

suivie de la même procédure forward-backward que dans l'entraînement Baum-Welch, avec les variables forward et backward :

$$\begin{aligned} \alpha_t(s) &= p(x_{1:t}, q_t = s|\mathbf{W}) \\ \beta_t(s) &= p(x_{t+1:T}|q_t = s, x_{1:t}, \mathbf{W}) \end{aligned}$$

où $s \in \mathcal{Q}(\mathbf{W})$. Ces variables sont calculées itérativement :

$$\alpha_t(s) = \frac{p(q_t = s|x_t)}{p(s)} \times \sum_{r\in\mathcal{Q}(\mathbf{W})} \alpha_{t-1}(r) p(q_t = s|q_{t-1} = r, \mathbf{W})$$

$$\beta_t(s) = \sum_{r\in\mathcal{Q}(\mathbf{W})} \frac{p(q_{t+1} = r|x_{t+1})}{p(r)} p(q_{t+1} = r|q_t = s, \mathbf{W}) \beta_{t+1}(r)$$

Puisque $\frac{p(q|x)}{p(q)} = \frac{p(x|q)}{p(x)}$, cette procédure calcule en fait :

$$\frac{p(\mathbf{x}|\mathbf{W})}{\prod_{t=1}^{|\mathbf{x}|} p(x_t)} = \sum_{s \in \mathcal{Q}(\mathbf{W})} \alpha_t(s)\beta_t(s) \qquad [2]$$

et on peut en déduire les probabilités a posteriori des états sachant $\mathbf{W}$:

$$p(q_t = s \in \mathcal{Q}(\mathbf{W})|\mathbf{x}, \mathbf{W}) = \frac{\alpha_t(s)\beta_t(s)}{\sum_r \alpha_t(r)\beta_t(r)}$$

En sommant sur toutes les occurrences d'un état donné dans le MMC de la séquence de mots, on obtient la probabilité a posteriori d'un identifiant d'état donné :

$$p(q_t = k \in \mathcal{Q}^*|\mathbf{x}, \mathbf{W}) = \sum_{s:\mu(s)=k} p(q_t = s|\mathbf{x}, \mathbf{W}) \qquad [3]$$

A chaque instant t, le réseau calcule une distribution a posteriori sur les éléments de $\mathcal{Q}^*$, si bien que la distribution de l'équation 3 peut être utilisé dans un coût d'entropie croisée pour le réseau de neurones. L'erreur propagée est

$$\frac{\partial E}{\partial a_k^t} = y_k^t - p(q_t = k \in \mathcal{Q}^*|\mathbf{x}, \mathbf{W}) = y_k^t - \sum_{s:\mu(s)=k} \frac{\alpha_t(s)\beta_t(s)}{\sum_r \alpha_t(r)\beta_t(r)} \qquad [4]$$

où y_k^t est la sortie du réseau à t pour l'identifiant k, *i.e.* $p(q_t = k|x_t)$, et a_k^t sont les activations avant la softmax.

Plusieurs articles ont été publiés concernant l'entraînement forward-backward des réseaux de neurones. L'idée est de remplacer la segmentation explicite de la séquence d'entrée avec l'algorithme Viterbi, par la prise en compte de toutes les alternatives de segmentation, comme c'est le cas dans l'algorithme de Baum-Welch pour les MMCs. Certain travaux (Senior et Robinson, 1996 ; Yan *et al.*, 1997) font l'hypothèse que $\frac{p(q|x)}{p(q)} \propto p(x|q)$, de sorte que les Equations 1 et 2 calculent toutes deux $p(\mathbf{x}|\mathbf{W})$. Les valeurs cibles pour l'entraînement découlent ensuite de l'équation 3. Partant d'autres hypothèses simplificatrices (Hennebert *et al.*, 1997 ; Konig *et al.*, 1996) arrivent aux mêmes résultats. Des articles plus anciens (Bengio *et al.*, 1992 ; Haffner, 1993) utilisent la procédure forward-backward pour entraîner les réseaux avec un critère de maximisation de l'information mutuelle (MMI).

Globalement, l'idée dans ces travaux est de remplacer les cibles du réseaux par des distributions plus lisses et prenant en compte toutes les segmentations possibles. Les réseaux sont déjà entraînés à partir des alignements Viterbi, puis les distributions sont estimées à l'aide de ces réseaux avec la procédure forward-backward, et l'entraînement se poursuit avec ces nouvelles cibles fixées. Il est possible de montrer qu'en utilisant un critère d'entropie croisée, et en ré-estimant les distributions après chaque itération, la méthode de (Senior et Robinson, 1996 ; Yan *et al.*, 1997) équivaut à entraîner le réseau à minimiser la log-vraisemblance $-\log p(\mathbf{x}|\mathbf{W})$.

2.3. *Les équations de l'entraînement CTC des réseaux récurrents*

Le but de la CTC (Graves *et al.*, 2006) est d'utiliser un réseau de neurones pour transformer une séquence $\mathbf{x}$ en une (plus courte) séquence de labels $\mathbf{L}$ (*e.g.* une séquence de caractères), sans recourir à un post-traitement compliqué. Dans cette approche, les sorties du réseau sont les différents labels, plus une sortie pour un label spécial appelé *blank* ($\oslash$), de sorte qu'un mapping très simple transforme une séquence de labels prédits en une séquence de sortie, obtenue en supprimant d'abord les répétitions de labels, puis les *blanks*. Par exemple :

$$a\ a \oslash \oslash b\ b \oslash ba \mapsto abba$$

L'une des motivations pour ce label spécial est la possibilité avec cette fonction simple de prédire deux labels consécutifs et identiques (Graves *et al.*, 2006). De plus, les auteurs suggèrent qu'il pourrait servir à modéliser les parties moins pertinentes de l'entrée, comme les connexions entre caractères dans l'écriture cursive ou les petits espaces blancs.

Dans ce paradigme, le réseau est entraîné à maximiser la probabilité de la séquence de labels sachant la séquence d'entrée (*i.e.* minimiser $(-\log p(\mathbf{L}|\mathbf{x}))$, et que plusieurs séquences de prédictions correspondent à la même séquence de labels (*e.g.* $a\ a\ b\ b$, $a\ a\ a\ b$, $a \oslash b\ b$, ...). Afin de simplifier l'analogie avec les précédentes méthodes, on appelle $\mathcal{Q}_n(\mathbf{L})$ l'ensemble des séquence de labels de longueur n correspondant à $\mathbf{L}$. Ainsi

$$p(\mathbf{L}|\mathbf{x}) = \sum_{\mathbf{q} \in \mathcal{Q}_{|\mathbf{x}|}(\mathbf{L})} p(\mathbf{q}|\mathbf{x})$$

En posant l'hypothèse que les prédictions à différents instants sont indépendantes (Graves *et al.*, 2006),

$$p(\mathbf{L}|\mathbf{x}) = \sum_{\mathbf{q} \in \mathcal{Q}_{|\mathbf{x}|}(\mathbf{L})} \prod_{t=1}^{|\mathbf{x}|} p(q_t|\mathbf{x})$$

Cette quantité est aussi calculée de façon efficace par une procédure forward-backward. Les "transitions" sont définies par la fonction de mapping (on peut continuer à prédire le même label, ou prédire le suivant dans la séquence cible, ou un *blank*). Les variables forward et backward sont définies comme suit, avec $\mathbf{L} = l_1 \dots l_n$ et $\mathbf{L}' = l'_1 \dots l'_n = \oslash l_1 \oslash \dots \oslash l_n \oslash$

$$\begin{aligned} \alpha_t(l'_s) &= p(q_{1:t} \in \mathcal{Q}_t(\mathbf{L}_{1:s/2}), q_t = l'_s|\mathbf{x}) \\ \beta_t(l'_s) &= p(q_{t+1:T} \in \mathcal{Q}_{T-t}(\mathbf{L}_{s/2+1:|\mathbf{L}|}), q_t = l'_s|\mathbf{x}) \end{aligned}$$

et les récurrences :

$$\alpha_t(l'_s) = p(q_t = l'_s|\mathbf{x}) \sum_{n=0}^{k} \alpha_{t-1}(l'_{s-n})$$

$$\beta_t(l'_s) = \sum_{n=0}^{k} p(q_{t+1} = l'_{s+n}|\mathbf{x})\beta_{t+1}(l'_{s+n})$$

avec $k = 1$ quand $l'_s = \varnothing$, ou quand $l'_s = l'_{s-2}$ (*resp.* $l'_s = l'_{s+2}$), et $k = 2$ sinon.

Les variables α et β permettent de calculer

$$p(\mathbf{L}|\mathbf{x}) = \sum_{\mathbf{q} \in \mathcal{Q}_{|\mathbf{x}|}(\mathbf{L})} \alpha_t(q)\beta_t(q)$$

et la dérivée de la fonction de coût $-\log p(\mathbf{L}|\mathbf{x})$ amène à l'erreur rétropropagée suivante :

$$\frac{\partial E}{\partial a_k^t} = y_k^t - \sum_{s:\mu(s)=k} \frac{\alpha_t(s)\beta_t(s)}{\sum_r \alpha_t(r)\beta_t(r)} \quad [5]$$

où y_k^t est la sortie du réseau à t pour le label k, et a_k^t sont les activations avant softmax.

2.4. *Similarités entre la CTC et l'entraînement des hybrides NN/MMC*

On note que l'équation 4 et l'équation 5 sont identiques. La différence se situe dans la façon dont y_k^t, α et β sont calculées. Dans la CTC, $y_k^t = p(q_t = k|\mathbf{x})$ plutôt que $p(q_t = k|x_t)$, mais cela est surtout dû au fait que la CTC est utilisée avec des réseaux récurrents. Les hypothèses d'indépendance dans (Hennebert *et al.*, 1997) notamment permettent de conclure que la CTC peut être applicable à n'importe quel réseau de neurones, et pas uniquement aux RNNs.

Quant aux variables forward et backward, mises à part les limites des sommes aux transitions autorisées dans la CTC, la différence principale est l'absence de probabilités de transition et de probabilités a priori pour la CTC. Si, dans la formulation présentée Section 2.2, on pose $\frac{p(r|s)}{p(r)} = 1$ quand une transition de s à r existe et 0 sinon, on obtient les équations de la CTC. Notons également que ces transitions ne sont pas prises en compte non plus dans l'entraînement au niveau trame.

Ainsi, on peut voir la CTC comme une simplification de l'entraînement forward-backward des hybrides NN/MMC, utilisant une topologie élémentaire de MMC où chaque caractère est représenté par un seul état, plus un état optionnel entre les caractères (correspondant au *blank*).

Si on ajoute à ces observations le fait que nous utilisons les RNNs entraînés avec la CTC dans des hybrides NN/MMC, on peut formuler les observations suivantes. L'utilisation de MMC supprime le besoin d'un post-traitement simple des sorties de réseau,

donc on n'est plus limité à une sortie par caractère. Le *blank* comme séparateur de caractères identiques consécutif n'est plus justifié non plus. Les équations de la CTC peuvent en fait être appliquées à n'importe quelle topologie de MMC. A l'inverse, la topologie induite par la CTC permet d'obtenir de bons résultats avec les RNNs. Le faible nombre d'états (ou de sorties de réseau) peut être avantageux (pour la taille du graphe de décodage, ou pour rendre la séparation des classes par le réseau plus facile). Nous pouvons donc essayer d'entraîner des systèmes hybrides avec moins d'états de MMC, et étudier les bénéfices de l'introduction d'un modèle de non-caractère (*blank*) dans ces systèmes. Enfin, nous avons déjà vu que la CTC n'est pas limitée aux RNNs, et pourrait être appliquée à d'autres types de réseaux.

Nous étudions ces différents aspects dans la suite de l'article, à savoir l'influence du nombre d'états, du label *blank*, et du critère d'entraînement (CTC et trame), ainsi que la façon dont ces aspects interagissent.

3. Entrainements trame et CTC : influence du type de modèle, du nombre d'états et du *blank*

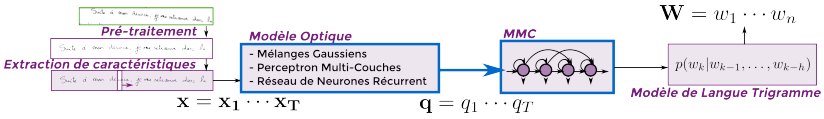

Figure 1 – Système global étudié.

Afin d'étudier ces différents aspects, nous avons entraîné des MMC "classiques" basés sur des modèles à mélanges gaussiens (GMMs), des perceptrons multicouches (MLPs) et des RNNs, pour les intégrer dans une chaîne de reconnaissance complète standard (Figure 1). Les expériences ont été menées sur la base de données publique IAM (Marti et Bunke, 2002) contenant des lignes de texte manuscrit en anglais. La base comprend 747 documents (6,482 lignes) pour l'entraînement, 116 (976) pour la validation, et 336 (2,915) pour le test. Les images sont d'abord prétraitées en corrigeant l'inclinaison des lignes, en augmentant le contraste et en normalisant la hauteur des lignes à 72 pixels. Des séquences de vecteurs de 56 caractéristiques géométriques et statistiques (Bianne *et al.*, 2011) sont extraits à l'aide d'une fenêtre glissante de taille 3px. En moyenne, nous avons 12 trames par caractère. Pour le décodage, nous avons utilisé un modèle de langue de type trigramme, estimé sur les corpus LOB, Wellington et Brown, limité au 50 000 mots les plus fréquents. Les résultats sont présentés sur l'ensemble de validation, où le modèle de langue présente un taux de mots hors vocabulaire de 4.3% et une perplexité de 298.

L'entraînement des GMMs suit la procédure EM standard, en réalignant les données et en mettant à jour les mélanges gaussiens à chaque itération, jusqu'à ce qu'aucune amélioration ne soit observée en validation. Les architectures de réseaux de neurones résultent d'un compromis entre taille (*i.e.* temps d'entraînement) et performance. Le but de ces expériences n'est pas d'obtenir les meilleurs résultats absolus,

ni de déterminer le meilleur nombre d'états de MMC, mais plutôt d'étudier, d'évaluer et de comparer les tendances sur les taux d'erreurs quand on varie la topologie et la méthode d'entraînement. Nous nous sommes donc limités à de petits réseaux. Les MLPs ont deux couches cachées de 1 024 neurones sigmoïdes chacune. Leurs entrées sont la concaténation de 11 trames.[1] Les RNNs ont une couche cachée LSTM de 100 neurones dans deux directions de récurrences (gauche-droite et droite-gauche). Leurs entrées sont directement les séquences de vecteurs de caractéristiques.

3.1. *Influence de la topologie et du blank*

Nous avons tout d'abord entraîné des systèmes standard GMM/MMC. Chaque état a son propre modèle d'émission (*i.e.* jeu de paramètres). Nous souhaitons confirmer qu'utiliser plusieurs états par caractère était meilleur qu'un ou deux pour ce modèle, et voir dans quelle mesure un modèle de *blank* pour les inter-caractères serait bénéfique. Les résultats sont présentés dans le Tableau 1. On observe qu'ajouter des états améliore les résultats. En comparant les deux lignes, on note aussi qu'ajouter un modèle de *blank* aide, mais pas autant qu'ajouter un état propre au caractère.

Tableau 1 – Taux d'erreur mot (caractère, %) de différents systèmes avec différentes topologies de MMC. Les RNNs sont entrainés avec la CTC, les MLPs au niveau trame, et les GMM/HMMs avec EM-Viterbi.

États	**1**	**2**	**3**	**4**	**5**	**6**	**7**
GMM							
No *blank*		25.7 (15.5)	20.8 (10.7)	17.3 (8.2)	16.7 (7.7)	16.5 (7.4)	**16.3 (6.9)**
Blank	30.1 (18.0)	23.5 (12.6)	18.3 (8.7)	17.1 (7.7)	17.3 (7.4)	17.0 (7.4)	18.7 (8.6)
MLP							
No *blank*		17.8 (8.2)	15.0 (6.1)	13.6 (5.3)	13.2 (4.8)	**12.4 (4.6)**	14.8 (4.8)
Blank	19.6 (9.0)	16.0 (6.3)	14.4 (5.5)	14.1 (5.2)	13.9 (5.2)	14.3 (5.9)	16.0 (6.7)
RNN							
No *blank*		19.3 (8.0)	16.5 (6.1)	14.1 (5.3)	13.7 (5.0)	14.1 (4.9)	14.5 (5.1)
Blank	**13.1 (4.9)**	13.9 (5.0)	14.3 (5.2)	13.9 (5.1)	14.9 (5.4)	15.3 (5.8)	14.2 (5.4)

Les MLPs sont entraînés au niveau trame avec un critère d'entropie croisée, qui se concentre sur la classification des trames individuelles (sans aspect séquentiel). Les classes cibles pour l'entraînement sont obtenues par alignement forcé (Viterbi) des données en utilisant les GMM/MMCs. Les résultats sont présentés sur les deux lignes du milieu du Tableau 1. Mise à part l'amélioration significative due à l'entraînement discriminant, nous tirons deux conclusions concernant le nombre d'états. Ajouter des états aux modèles de caractères améliore les résultats, tandis que l'ajout d'un modèle *blank* n'aide que quand les modèles de caractères sont petits.

1. Ce choix est le fruit d'expériences avec un entraînement au niveau trame et six états par caractère, qui dépassent le cadre de cet article (se référer à (Bluche, 2015)). Nous n'avons pas réajusté ce chiffre pour chacune des topologies testées ici.

Les RNNs sont entraînés avec le critère CTC, mais sans les contraintes d'une sortie par caractère et pour le *blank*. Comme on utilise les RNNs dans un système hybride, les sorties sont en fait des états de MMC, comme pour les MLPs. On applique donc la CTC avec le modèle de transition défini par ces MMCs. Les résultats sont présentés dans les deux dernières lignes du Tableau 1. Sans *blank*, l'ajout d'états aux modèles de caractères améliore encore les résultats. Comme pour les MLPs, l'ajout du *blank* n'aide que quand les modèles sont petits.

Il y a cependant une différence notable avec les résultats des MLPs : pour les RNNs avec le *blank*, l'ajout d'état dégrade les résultats. Dans les modèles précédents, la topologie "CTC" donnait les moins bons résultats, alors qu'ici elle semble être le meilleur choix. Il convient donc ici de se demander si cela est dû à la CTC ou à la récurrence du réseau, voire à une combinaison des deux. Nous tentons d'y apporter une réponse dans les sections suivantes. Ceci dit, il semble clair que le rôle de ce label spécial n'est pas uniquement de modéliser les inter-caractères.

3.2. *Entrainement CTC de perceptrons multicouches*

Nous avons vu que les ressemblances entre la CTC et l'entraînement d'hybrides NN/MMCs permettaient d'envisager l'utilisation de cette méthode avec d'autres modèles que les RNNs, et avec des topologies différentes. Nous l'avons donc appliquée à des MLPs, avec la topologie "CTC" comportant un état par caractère et le *blank*, et avec la topologie "MMC" à six états sans *blank*.

Tableau 2 – Entrainement CTC de MLPs. La topologie *"MMC"* comporte six états par caractère, et la topologie *"CTC"* un état par caractère, et un modèle de *blank*.

Entrainement	**Topologie**	**Erreurs mot (%)**	**Erreurs caractère (%)**
Trame	MMC	12.4	4.6
CTC	MMC	12.6	4.3
	CTC	17.6	7.4

Les résultats sont présentés dans le Tableau 2, et comparés au meilleur MLP entraîné au niveau trame. Avec la topologie à six états sans *blank* ("MMC"), on n'observe pas de différence significative entre entraînement trame et CTC. En revanche, avec la topologie "CTC", les résultats deviennent bien pires. Ce choix de topologie pour l'entraînement CTC, qui était le meilleur avec les RNNs, ne semble pas adapté aux MLPs. Il faut néanmoins noter que les taux d'erreurs pour cette topologie sont toutefois plus bas qu'avec l'entraînement au niveau trame (19.6% / 9.0%, Tableau 1).

3.3. *Comparaison expérimentale de l'entraînement au niveau trame et CTC*

Dans cette section, nous complétons la comparaison en entraînant les deux types de modèle avec les deux méthodes d'entraînement, pour différentes topologies avec

et sans *blank*. Dans la littérature, ce genre de comparaison se limite à la topologie "MMC" pour l'entraînement au niveau trame et à la topologie à un état avec *blank* pour la CTC (Graves *et al.*, 2006 ; Morillot *et al.*, 2013). (Maas *et al.*, 2014) comparent des réseaux profonds avec ou sans récurrence entraînés avec la CTC, et obtiennent de bien meilleurs résultats avec, ce que nous allons confirmer. Ici, en comparant les deux modèles sous toutes les variations de topologie et d'entraînement, nous pouvons mettre en évidence l'effet des différents aspects pour tenter de comprendre la relation entre topologie, *blank*, modèle et méthode d'entraînement.

Les résultats sont reportés sur la Figure 2, où, pour chaque modèle, nous affichons le taux d'erreur en fonction du nombre d'états pour la CTC (violet) et le niveau trame (bleu), avec *blank* (pointillés) ou sans (traits pleins). On voit que les systèmes avec *blank* sont meilleurs lorsqu'il y a peu d'états, et moins bons avec plus d'états. Sans *blank*, la différence de performance entre les méthodes d'entraînement est marginale.

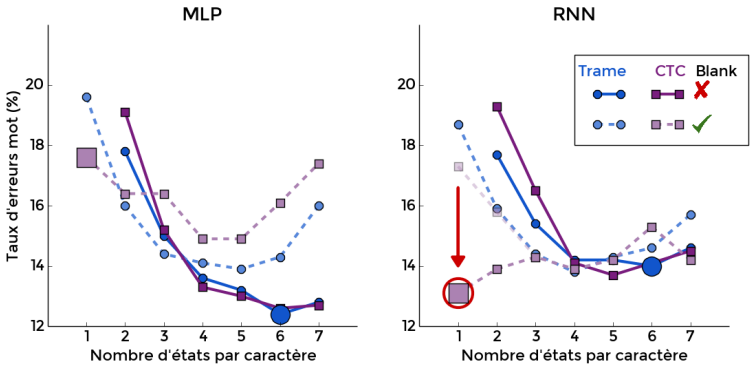

Figure 2 – Comparaison de taux d'erreurs caractère avec l'entraînement au niveau trame et CTC, avec ou sans *blank*, pour différents nombres d'états par caractère. (à gauche : MLPs ; à droite : RNNs).

De plus, on observe que toutes les courbes ont la même forme : le taux d'erreur diminue quand on augmente le nombre d'états, et commence à augmenter quand ce nombre devient trop grand, et cela se produit plus tôt avec le label *blank*. Un seul cas échappe à cette tendance : celui du RNN entraîné avec la CTC et le *blank*, qui donne de particulièrement bons résultats avec peu d'états, contrairement à ce dont on pourrait attendre en voyant les autres cas (envisagé par la partie transparente hypothétique). Cela suggère que la CTC, avec un seul état, le *blank*, et la procédure forward-backward, est particulièrement adaptée aux RNNs, et que ces trois aspects ne fonctionnent bien que quand ils sont utilisés ensemble.

Enfin, précisons que pour obtenir une convergence de la CTC pour les modèles sans *blank*, nous avons dû effectuer une première époque d'entraînement au niveau trame pour initialiser les réseaux, sans quoi les taux d'erreurs dépassaient les 90%, et les réseaux ne prédisaient pratiquement que des espaces. Ce problème n'apparaît plus

quand on introduit le *blank*, ce qui suggère que ce label joue un rôle crucial dans la procédure d'alignement au début de l'entraînement CTC.

4. Interactions entre l'entraînement CTC et le label *blank*

4.1. *Observation de pics dans les prédictions*

Il s'avère assez typique pour les RNNs entraînés avec la CTC d'observer des prédictions complètement dominées par le *blank*, avec des pics de prédiction très localisés pour les caractères. Il est intéressant de voir que des phénomènes similaires se produisent avec plus d'états par caractère, mais également pour des MLPs.

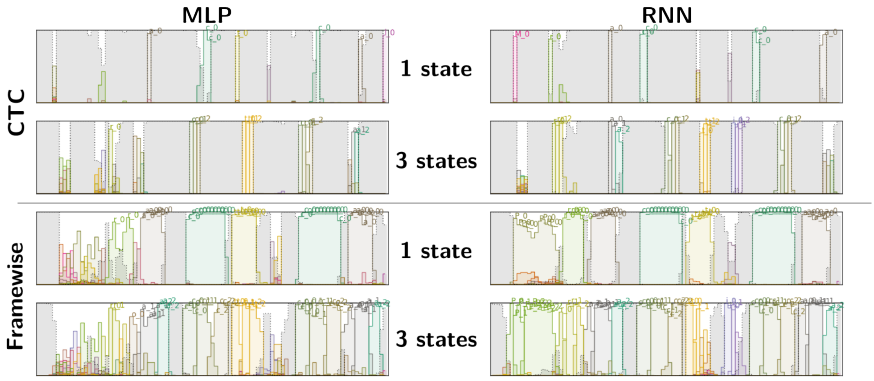

Figure 3 – Prédictions de réseaux de neurones avec des topologies de MMC et des méthodes d'entraînement différentes. Chaque figure représente les probabilités a posteriori des réseaux à différents instants. La sortie du *blank* est affichée en gris, les autres sorties (caractères / états de MMC) en couleurs.

Nous illustrons cela sur la Figure 3, qui montre des séquences de prédictions de réseaux. Chaque caractère ou état de MMC et affiché en couleur, et les *blanks* en gris. On note également que ces pics ne sont pas visibles lors de l'entraînement au niveau trame. On peut donc en conclure que ce phénomène n'est dû ni au RNN, ni aux modèles à un état par caractère, mais plutôt à l'interaction du *blank* et de l'entraînement CTC.

La Figure 4 monte l'évolution des prédictions du RNN à un état avec *blank* au cours de l'entraînement CTC. On voit qu'au début de l'entraînement, les prédictions sont plutôt uniformes, et que le réseau commence par apprendre à ne prédire que des *blanks*. Une fois qu'il sait prédire le *blank* avec une forte probabilité, il commence à apprendre à prédire les caractères, et des pics apparaissent à des endroits très précis.

Cela n'est en fait pas très surprenant. Dans le calcul du coût de la CTC, il y a un *blank* possible entre chaque caractère. Au début de l'entraînement, la distribution en

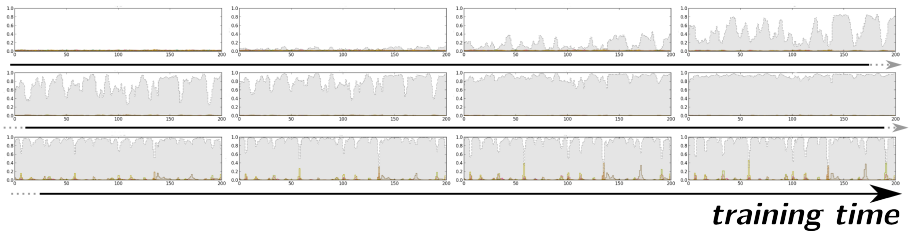

Figure 4 – Evolution des sorties de RNN pour une ligne donné pendant l'entraînement CTC. Comme dans la Figure 3, le gris correspond au *blank*, et les couleurs aux différents caractères

sortie du réseau est plus ou moins uniforme, et tous les chemins menant à la transcription correcte ont à peu près la même probabilité. Cependant, quand on additionne les probabilités a posteriori des différentes occurrences du *blank*, on obtient une probabilité cible bien plus forte pour ce label que pour tous les autres, ce qui aura un effet sur l'erreur rétropropagée dans le réseau. Ce signal incite le réseau à attribuer une plus grande probabilité au *blank* en général, ce qui en retour donnera plus de poids aux segmentations comportant une majorité de *blanks* aux prochaines itérations.

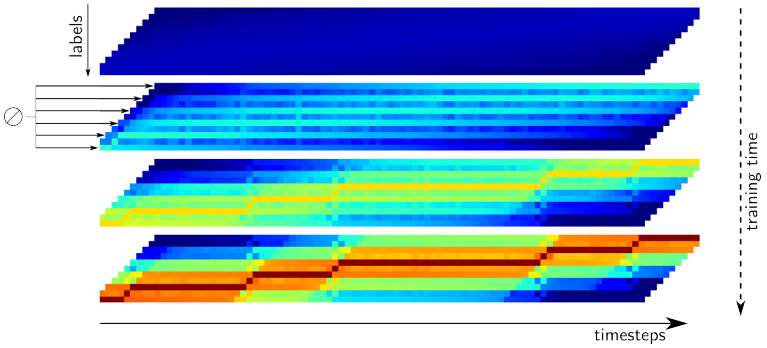

Figure 5 – Visualisation des probabilités a posteriori calculées avec la procédure forward-backward pendant l'entraînement CTC.

Sur la Figure 5 nous illustrons les probabilités a posteriori calculées par la procédure forward-backward au cours de l'entraînement, qui constituent le signal cible pour le réseau. Une ligne sur deux correspondant au *blank*, on comprend comment les chemins avec plus de *blank* deviennent prépondérants.

Cependant, ne prédire que des *blanks* va d'une part diminuer la force de l'erreur rétropropagée, et va nuire au coût, à cause des trop faibles probabilités des caractères. Pour pallier à ce problème, le réseau n'a qu'à trouver une position pour passer d'une séquence de *blanks* à une autre, et cette position va être renforcée pour les mêmes

raisons que celles qui rendent le *blank* dominant. Finalement, le réseau se concentre assez rapidement sur un nombre très limité d'hypothèses de segmentation, et le mécanisme même de la CTC, quand le *blank* est présent, favorise les solutions où le *blank* est prédit partout sauf à des positions bien précises.

Le fait que la CTC formulée de la sorte soit plus adaptée aux RNNs qu'aux MLPs est donc certainement dû à la capacité des RNNs à considérer un contexte large pour prédire ces pics, quand les MLPs n'ont accès qu'à un contexte limité à chaque instant.

4.2. *Les advantages de ces pics*

A part l'impossibilité d'obtenir une segmentation précise en caractères, ces pics ne présentent pas nécessairement d'inconvénients pour le décodage avec modèle de langue. En effet, les graphes de décodage avec un état par caractère sont beaucoup plus compacts. De plus, bien qu'il y ait moins d'ambiguïté dans les sorties du réseau, comme le *blank* est partagé entre tous les modèles de mots, le nombre de prédictions à changer pour passer d'un mot erroné à un mot correct est bien plus faible, ce qui rend les erreurs moins coûteuses au décodage, en plus de permettre de garder plus d'hypothèses de mots lors de la recherche dans le graphe. De façon assez intéressante, nous avons observé que le facteur optimal, qui définit l'importance relative donnée au réseau de neurones lors du décodage par rapport au modèle de langue, est inversement proportionnel au nombre de prédictions nécessaires pour reconnaître un caractère. Il était entre 10^{-1} et 15^{-1} pour les systèmes MMC classiques, et N^{-1} pour les systèmes avec CTC et *blank*, et N états par caractère.

4.3. *Faut-il essayer d'éviter ces pics ?*

L'analyse précédente nous permet de mettre en place des expériences pour essayer d'éviter les pics de prédiction observés, ou pour tenter d'améliorer les modèles en tirant parti de la compréhension de l'entraînement CTC.

Initialisation au niveau trame Nous avons vu que les pics n'étaient pas visibles lors de l'entraînement au niveau trame, et qu'une initialisation au niveau trame permettait de faire converger les modèles sans *blank*. Nous avons donc effectué une époque d'entraînement au niveau trame avant la CTC, mais il s'avère avec les RNNs que les pics réapparaissent quand on commence l'entraînement CTC.

Répétition de labels pendant l'entraînement L'omniprésence du *blank* par rapport aux autres labels dans le calcul de la CTC provoque l'apparition de pics. Pour limiter le phénomène, nous avons forcé lors de l'entraînement la CTC à passer par au moins 2, 3 ou 5 répétitions du label. Cela correspondrait à un MMC à 2, 3 ou 5 états identiques. En revanche, cette contrainte disparaît au décodage. Les pics disparaissent effectivement, mais nous avons observé que le RNN apprend en fait à reconnaître le caractère sur un durée d'exactement 2, 3 ou 5 trames, sans variation, et le reste des prédictions est toujours composé de *blanks*.

Pénalité sur le *blank* Nous avons également tenté d'ajouter une pénalité dans le calcul de la CTC pour les chemins passant par des *blanks*. En faisant varier la valeur de cette pénalité, on arrive effectivement à contrôler l'importance des pics, et à les rendre plus larges. Cependant, on finit par retrouver les mauvaises performances observées pour les systèmes avec peu d'états et sans *blanks*.

Diminution du learning rate Afin de diminuer l'effet des forts gradients sur le *blank* au début de l'entraînement, et garder plus longtemps plus d'alternatives de segmentation, nous avons entraîné les réseaux avec un learning rate plus petit. Des exemples de résultats sont présentés sur la Figure 6. Pour les RNNs, les prédictions

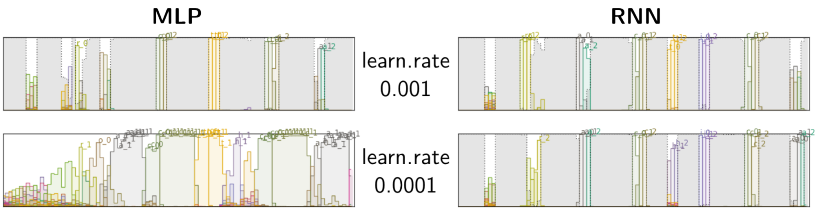

Figure 6 – Effet de la diminution du learning rate sur les prédictions de réseaux entrainés avec la CTC.

sont toujours des pics, ce qui semble confirmer que la meilleure option pour eux est d'utiliser leur récurrence pour donner des prédictions localisées. Les performances sont cependant moins hautes. En revanche, nous sommes parvenus à éliminer les pics dans le cas des MLPs, tout en obtenant de meilleures performances, ce qui confirme qu'il est plus difficile et sous-optimal pour un MLP de prédire ces pics localisés.

Learning rate adaptatif par paramètre (RMSProp) Sur des problèmes plus compliqués que la base IAM (par exemple, MAURDOR (Brunessaux *et al.*, 2014)), il arrive d'observer lors de l'apprentissage avec la CTC un phénomène de plateau, où la valeur du coût stagne pendant assez longtemps à une valeur qui correspond à la prédiction de *blanks* uniquement (voir les courbes noires et bleues de la Figure 7). (Louradour et Kermorvant, 2014) l'expliquent par la difficulté pour cette méthode d'entraînement d'aligner et de reconnaître les caractères à la fois, et proposent un curriculum d'apprentissage, où les séquences courtes pour lesquelles le problème est plus facile sont privilégiées au début de l'entraînement.

Au regard de notre analyse, nous supposons que cela est aussi lié à la difficulté du réseau à commencer à prédire des caractères plutôt que le *blank*, à cause du signal d'erreur rétropropagé. Dans la lignée de l'expérience précédente, nous avons choisi une méthode d'adaptation du learning rate par paramètre : RMSProp (Tieleman et Hinton, 2012). Cette approche consiste à normaliser le gradient pour un paramètre donné en fonction de l'intensité des gradients observés par le passé. Ainsi, le learning rate effectif sera petit si le paramètre à eu des gradients fort, et grand dans le cas contraire. En raisonnant du point de vue des labels, on peut espérer que la méthode atténuera les gradients importants venant du *blank* pour privilégier l'apprentissage des caractères.

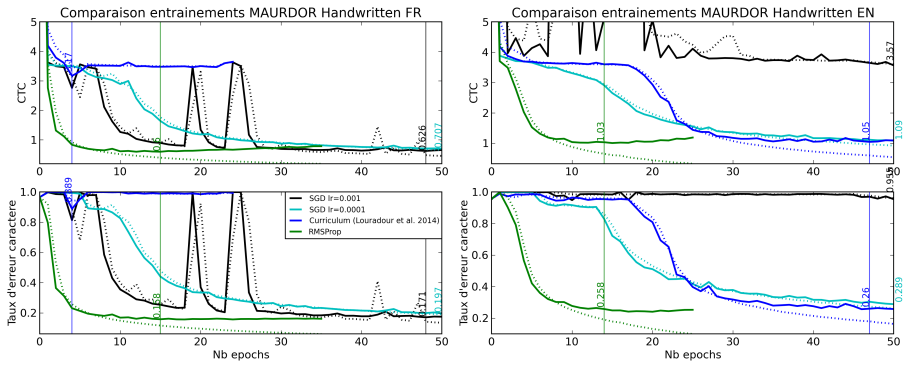

Figure 7 – L'utilisation de RMSProp (en vert) permet d'éliminer le plateau au début de l'apprentissage CTC et donne une convergence rapide des modèles.

Bien qu'au final les pics sont toujours présents, on voit sur la Figure 7 que cette approche permet en effet de résoudre ce problème, et permet une convergence beaucoup plus rapide des apprentissages qu'avec une descente de gradient stochastique (SGD) simple ou que l'apprentissage par curriculum.

5. Conclusion

Dans cet article, nous nous sommes intéressés à l'entraînement CTC de réseaux de neurones, permettant d'apprendre à étiqueter des séquences sans segmentation préalable. Tout d'abord, une étude théorique nous a permis de voir que l'entraînement CTC était très similaire à l'entraînement de systèmes hybrides NN/MMC. On peut voir la CTC comme la simplification d'un cas particulier de topologie à un état avec un modèle spécial pour le label *blank*. Ces observations nous ont amené à entraîner des MLPs avec la CTC, et à comparer l'entraînement au niveau trame à la CTC, en faisant varier le nombre d'états de MMC ainsi que la présence de ce label spécial et intrigant. Nous avons pu ainsi mettre en évidence l'interaction particulièrement bonne entre ce label, la CTC et les RNNs. Enfin, nous avons poussé plus loin cette analyse dans un dernier temps pour tenter d'apporter une explication à ce comportement et pour mieux comprendre le phénomène. Cela nous a finalement permis de choisir des méthodes plus adaptées pour l'entraînement CTC des RNNs.

6. Bibliographie

Bengio Y., De Mori R., Flammia G., Kompe R., « Global optimization of a neural network-hidden Markov model hybrid », *Neural Networks, IEEE Transactions on*, vol. 3, n° 2, p. 252-259, 1992.

Bianne A.-L., Menasri F., Al-Hajj R., Mokbel C., Kermorvant C., Likforman-Sulem L., « Dynamic and Contextual Information in HMM modeling for Handwriting Recognition », *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, nº 10, p. 2066 - 2080, 2011.

Bluche T., Deep Neural Network for Large Vocabulary Handwritten Text Recognition, PhD thesis, Université Paris-Sud, 2015.

Brunessaux S., Giroux P., Grilhères B., Manta M., Bodin M., Choukri K., Galibert O., Kahn J., « The Maurdor Project : Improving Automatic Processing of Digital Documents », *Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on*, IEEE, p. 349-354, 2014.

Graves A., Fernández S., Gomez F., Schmidhuber J., « Connectionist temporal classification : labelling unsegmented sequence data with recurrent neural networks », *International Conference on Machine learning*, p. 369-376, 2006.

Haffner P., « Connectionist speech recognition with a global MMI algorithm. », *EUROSPEECH*, 1993.

Hennebert J., Ris C., Bourlard H., Renals S., Morgan N., « Estimation of global posteriors and forward-backward training of hybrid HMM/ANN systems. », 1997.

Konig Y., Bourlard H., Morgan N., « Remap : Recursive estimation and maximization of a posteriori probabilities-application to transition-based connectionist speech recognition », *Advances in Neural Information Processing Systems*, vol. , p. 388-394, 1996.

Louradour J., Kermorvant C., « Curriculum Learning for Handwritten Text Line Recognition », *International Workshop on Document Analysis Systems (DAS)*, 2014.

Maas A. L., Hannun A. Y., Jurafsky D., Ng A. Y., « First-Pass Large Vocabulary Continuous Speech Recognition using Bi-Directional Recurrent DNNs », *arXiv preprint arXiv :1408.2873*, 2014.

Marti U.-V., Bunke H., « The IAM-database : an English sentence database for offline handwriting recognition », *International Journal on Document Analysis and Recognition*, vol. 5, nº 1, p. 39-46, 2002.

Morillot O., Likforman-Sulem L., Grosicki E., « Comparative study of HMM and BLSTM segmentation-free approaches for the recognition of handwritten text-lines », *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, IEEE, p. 783-787, 2013.

Rabiner L., Juang B.-H., « An introduction to hidden Markov models », *ASSP Magazine, IEEE*, vol. 3, nº 1, p. 4-16, 1986.

Senior A., Robinson T., « Forward-backward retraining of recurrent neural networks. », *Advances in Neural Information Processing Systems*, vol. , p. 743-749, 1996.

Tieleman T., Hinton G., « Lecture 6.5-rmsprop : Divide the gradient by a running average of its recent magnitude », *COURSERA : Neural Networks for Machine Learning*, 2012.

Tong A., Przybocki M., Maergner V., El Abed H., « NIST 2013 Open Handwriting Recognition and Translation (OpenHaRT13) Evaluation », *11th IAPR Workshop on Document Analysis Systems (DAS2014)*, 2014.

Yan Y., Fanty M., Cole R., « Speech recognition using neural networks with forward-backward probability generated targets », *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, vol. 4, IEEE Computer Society, p. 3241-3241, 1997.