Equations différentielles ordinaires neurales

Ricky TQ Chen *, Yulia Rubanova *, Jesse Bettencourt *, David Duvenaud
Université de Toronto, Vector Institute

Abstrait

Nous introduisons une nouvelle famille de modèles de réseaux de neurones profonds. Au lieu de spécifier une séquence discrète de couches cachées, nous paramétrons la dérivée de l'état caché à l'aide d'un réseau de neurones. La sortie du réseau est calculée à l'aide d'un solveur d'équations différentielles en boîte noire. Ces modèles à profondeur continue ont un coût de mémoire constant, adaptent leur stratégie d'évaluation à chaque entrée et peuvent explicitement échanger la précision numérique contre la vitesse. Nous démontrons ces propriétés dans des réseaux résiduels à profondeur continue et des modèles de variables latentes en temps continu. Nous construisons également des fl ux de normalisation continus, un modèle génératif qui peut s'entraîner par maximum de vraisemblance, sans partitionner ni ordonner les dimensions des données. Pour la formation, nous montrons comment rétropropagérer de manière évolutive via n'importe quel solveur ODE, sans accéder à ses opérations internes.

1. Introduction

Des modèles tels que les réseaux résiduels, les décodeurs de réseaux neuronaux récurrents et les fl ux de normalisation créent des transformations compliquées en composant une séquence de transformations vers un état caché:

$$h_{t+1} = h_t + F(h_t, \theta_t) \quad (1)$$

où $t \in \{0 \ldots T\}$ et $h_t \in R^{RÉ.}$ Ces mises à jour itératives peuvent être vues comme une discrétisation Euler d'un transformation continue (Lu et coll. , 2017 ; Haber et Ruthotto , 2017 ; Ruthotto et Haber , 2018).

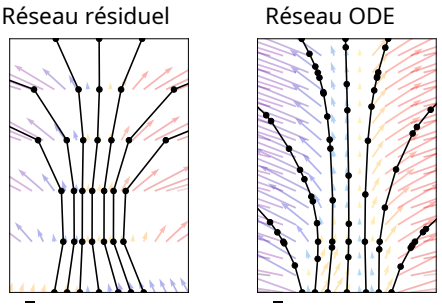


Figure 1: La gauche: Un réseau résiduel dé fi nit une séquence discrète de transformations fi nies. Droite: Un réseau ODE dé fi nit un champ vectoriel, qui transforme continuellement l'état. Tous les deux: Les cercles représentent les emplacements d'évaluation.

Que se passe-t-il lorsque nous ajoutons plus de couches et prenons des mesures plus petites? À la limite, nous paramétrons la dynamique continue des unités cachées à l'aide d'une équation différentielle ordinaire (ODE) spéci fi ée par un réseau de neurones:

$$\frac{dh(t)}{dt} = F(h(t), t, \theta) \quad (2)$$

À partir de la couche d'entrée $h(0)$, on peut dé fi nir la couche de sortie $h(T)$ être la solution à ce problème de valeur initiale ODE à un moment donné T. Cette valeur peut être calculée par un solveur d'équations différentielles en boîte noire, qui évalue la dynamique des unités cachées F le cas échéant, pour déterminer la solution avec la précision souhaitée. Chiffre 1 contraste ces deux approches.

La définition et l'évaluation de modèles à l'aide de solveurs ODE présentent plusieurs avantages:

Ef fi cacité de la mémoire Dans la section 2 , nous montrons comment calculer les gradients d'une perte scalaire par rapport à toutes les entrées de tout solveur ODE, sans rétropropagation à travers les opérations du solveur.
Ne pas stocker de quantités intermédiaires de la passe avant nous permet d'entraîner nos modèles avec un coût de mémoire constant en fonction de la profondeur, un goulot d'étranglement majeur de l'entraînement des modèles profonds.

32e Conférence sur les systèmes de traitement de l'information neuronale (NeurIPS 2018), Montréal, Canada.

Calcul adaptatif La méthode d'Euler est peut-être la méthode la plus simple pour résoudre les ODE. Depuis, il y a eu plus de 120 ans de développement de solveurs ODE efficaces et précis (Runge , 1895 ; Kutta , 1901 ; Hairer et coll. , 1987). Les solveurs ODE modernes fournissent des garanties sur la croissance de l'erreur d'approximation, surveillent le niveau d'erreur et adaptent leur stratégie d'évaluation à la volée pour atteindre le niveau de précision requis. Cela permet au coût d'évaluation d'un modèle de s'adapter à la complexité du problème. Après la formation, la précision peut être réduite pour les applications en temps réel ou à faible consommation.

Efficacité des paramètres Lorsque la dynamique des unités cachées est paramétrée comme une fonction continue du temps, les paramètres des «couches» proches sont automatiquement liés ensemble. Dans la section 3 , nous montrons que cela réduit le nombre de paramètres requis sur une tâche d'apprentissage supervisé.

Flux de normalisation évolutifs et inversibles Un avantage secondaire inattendu des transformations continues est que la formule de changement de variables devient plus facile à calculer. Dans la section 4 , nous dérivons ce résultat et l'utilisons pour construire une nouvelle classe de modèles de densité inversible qui évite le goulot d'étranglement mono-unité de normalisation des fl ux, et peut être entraînée directement par le maximum de vraisemblance.

Modèles de séries chronologiques continues Contrairement aux réseaux de neurones récurrents, qui nécessitent une discrétisation des intervalles d'observation et d'émission, la dynamique définie en continu peut naturellement incorporer des données qui arrivent à des moments arbitraires. Dans la section 5 , nous construisons et démontrons un tel modèle.

2 Différenciation automatique en mode inverse des solutions ODE

La principale difficulté technique dans la formation des réseaux en profondeur continue consiste à effectuer une différenciation en mode inverse (également connue sous le nom de rétropropagation) via le solveur ODE. La différenciation par les opérations de la passe avant est simple, mais entraîne un coût de mémoire élevé et introduit une erreur numérique supplémentaire.

Nous traitons le solveur ODE comme une boîte noire et calculons les dégradés à l'aide du méthode de sensibilité adjointe (Pontryagin et coll. , 1962). Cette approche calcule les gradients en résolvant une seconde ODE augmentée en arrière dans le temps et est applicable à tous les solveurs ODE. Cette approche évolue linéairement avec la taille du problème, a un faible coût de mémoire et contrôle explicitement l'erreur numérique.

Envisagez d'optimiser une fonction de perte à valeur scalaire L (), dont l'entrée est le résultat d'un solveur ODE:

$$L(z(t_1)) = L(z(t_0) + \int_{t_0}^{t_1} f(z(t), t, \theta)\, dt = L(\mathrm{ODESolve}(z(t_0), f, t_0, t_1, \theta)) \qquad (3)$$

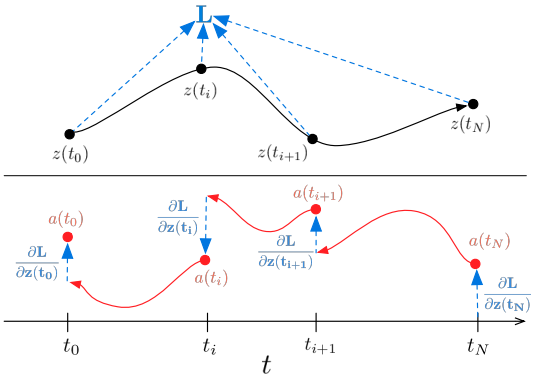


Figure 2: Différenciation en mode inverse d'une solution ODE. La méthode de sensibilité adjointe résout une ODE augmentée en arrière dans le temps. Le système augmenté contient à la fois l'état d'origine et la sensibilité de la perte par rapport à l'état. Si la perte dépend directement de l'état à plusieurs instants d'observation, l'état adjoint doit être mis à jour dans le sens de la dérivée partielle de la perte par rapport à chaque observation.

Optimiser L, nous avons besoin de gradients par rapport à θ. La première étape consiste à déterminer comment le gradient de la perte dépend de l'état caché z (t) à chaque instant. Cette quantité s'appelle le adjoint une(t) = $\partial L / \partial z(t)$. Sa dynamique est donnée par un autre ODE, qui peut être considéré comme l'analogue instantané de la règle de la chaîne:

$$\frac{\text{ré une(t)}}{dt} = -\,\text{une(t)}\,\frac{\partial f(z(t), t, \theta)}{\partial z} \qquad (4)$$

Nous pouvons calculer $\partial L / \partial z(t_0)$ par un autre appel à un solveur ODE. Ce solveur doit fonctionner à l'envers, à partir de la valeur initiale de $\partial L / \partial z(t_1)$. Une complication est que la résolution de cette ODE nécessite la valeur consciente de z (t) tout au long de sa trajectoire. Cependant, nous pouvons simplement recalculer z (t) en arrière dans le temps avec l'adjoint, à partir de sa valeur finale z (t 1).

Calcul des gradients par rapport aux paramètres θ nécessite d'évaluer une troisième intégrale, qui dépend des deux z (t) et une(t):

$$\frac{dL}{d\theta} = \int_{t_1}^{t_0} \text{une(t)}\,\frac{\partial f(z(t), t, \theta)}{\partial \theta}\, dt \qquad (5)$$

Les produits vector-jacobiens une(t) $^T \frac{\partial f}{\partial z}$ et une(t) $^T \frac{\partial f}{\partial \theta}$ dans (4) et (5) peut être évalué ef fi cacement par différenciation automatique, à un coût en temps similaire à celui de l'évaluation F. Toutes les intégrales pour la résolution z, a et $\frac{\partial L}{\partial \theta}$ peut être calculé en un seul appel à un solveur ODE, qui concatène l'état d'origine, le adjoint, et les autres dérivées partielles en un seul vecteur. Algorithme 1 montre comment construire la dynamique nécessaire et appeler un solveur ODE pour calculer tous les dégradés à la fois.

Algorithme 1 Dérivée en mode inverse d'un problème de valeur initiale ODE

Contribution: paramètres dynamiques θ, Heure de début t_0, temps d'arrêt t_1, état final z (t_1), gradient de perte $\partial L / \partial z(t_1)$

$s_0 = [\, z(t_1), \frac{\partial L}{\partial z(t_1)}, 0_{|\theta|}]$ — Définir l'état initial augmenté

def aug_dynamics ([z (t), une(t), ·], t, θ): — Dé fi nir la dynamique sur un état augmenté

  revenir [F(z (t), t, θ), - une(t) $^T \frac{\partial f}{\partial z}$, - une(t) $^T \frac{\partial f}{\partial \theta}$] — Calculer des produits vectoriels jacobiens

$[\, z(t_0), \frac{\partial L}{\partial z(t_0)}, \frac{\partial L}{\partial \theta}]$ = ODESolve (s_0, aug_dynamics, t_1, t_0, θ) — Résoudre l'ODE en temps inverse

revenir $\frac{\partial L}{\partial z(t_0)}, \frac{\partial L}{\partial \theta}$ — Dégradés de retour

La plupart des solveurs ODE ont la possibilité de sortir l'état z (t) à plusieurs reprises. Lorsque la perte dépend de ces états intermédiaires, la dérivée de mode inverse doit être décomposée en une séquence de résolutions séparées, une entre chaque paire consécutive de temps de sortie (Figure 2). A chaque observation, l'adjoint doit être ajusté dans le sens de la dérivée partielle correspondante $\partial L / \partial z(t_{je})$.

Les résultats ci-dessus étendent ceux de Stapor et coll. (2018 , section 2.4.2). Une version étendue de Algorithme 1 y compris les dérivés wrt t_0 et t_1 se trouve en annexe C . Les dérivations détaillées sont fournies en annexe B . appendice ré fournit du code Python qui calcule tous les dérivés pour

charge toutes les dérivées d'ordre supérieur. Depuis, nous avons publié un package de différenciation automatique PyTorch (Paszke et coll., 2017), y compris les implémentations basées sur GPU de plusieurs solveurs ODE standard à

.

3 Remplacement des réseaux résiduels par des ODE pour l'apprentissage supervisé

Dans cette section, nous étudions expérimentalement la formation des ODE neuronaux pour l'apprentissage supervisé.

Logiciel Pour résoudre numériquement les problèmes de valeur initiale ODE, nous utilisons la méthode implicite Adams implémentée dans LSODE et VODE et interfacée via le paquet. Étant une méthode implicite, elle a de meilleures garanties que les méthodes explicites telles que Runge-Kutta mais nécessite de résoudre un problème d'optimisation non linéaire à chaque étape. Cette configuration rend difficile la rétropropagation directe via l'intégrateur. Nous implémentons la méthode de sensibilité adjointe dans le cadre de Python (Maclaurin et coll. , 2015). Pour les expériences de cette section, nous avons évalué t h e h je ré ré e

n

la dynamique des états et leurs dérivés sur le GPU utilisant Tensor fl ow, qui ont ensuite été appelés depuis les solveurs Fortran ODE, qui ont été appelés depuis Python code.

Architectures de modèle Nous expérimentons avec un petit réseau résiduel qui sous-échantillonne l'entrée deux fois puis applique 6 blocs résiduels standard He et coll. (2016b), qui sont remplacés par un module ODESolve dans la variante ODE-Net. Nous testons également un réseau avec la même architecture mais où les gradients sont rétropropagés directement via un intégrateur Runge-Kutta, appelé RK-Net. Tableau 1 affiche l'erreur de test, le nombre de paramètres, et le coût de la mémoire. L dénote

Tableau 1: Performances sur MNIST. † De LeCun et coll. (1998)

	Erreur de test	# Paramètres	Mémoire	Temps
MLP †	1,60%	0,24 M	-	-
ResNet	0,41%	0,60 M	O (L)	O (L)
RK-Net	0,47%	0,22 M	O ($\tilde{L}$	O ($\tilde{L}$)
ODE-Net	0,42%	0,22 M	O)(1)	O ($\tilde{L}$)

le nombre de couches dans ResNet, et $\tilde{L}$ est le nombre d'évaluations de fonctions que le solveur ODE requêtes en une seule passe avant, qui peut être interprétée comme un nombre implicite de couches.

Nous constatons que les ODE-Nets et RK-Nets peuvent atteindre à peu près les mêmes performances que ResNet, tout en utilisant moins de paramètres. Pour référence, un réseau de neurones avec une seule couche cachée de 300 unités a à peu près le même nombre de paramètres que les architectures ODE-Net et RK-Net que nous avons testées.

Contrôle des erreurs dans les réseaux ODE Les solveurs ODE peuvent garantir approximativement que la sortie est dans une tolérance donnée de la vraie solution. La modification de cette tolérance modifie le comportement du réseau. Nous vérifions d'abord que l'erreur peut bien être contrôlée dans la figure 3 une. Le temps passé par l'appel de renvoi est proportionnel au nombre d'évaluations de fonction (Figure 3 b), donc le réglage de la tolérance nous donne un compromis entre la précision et le coût de calcul. On pourrait s'entraîner avec une grande précision, mais passer à une précision inférieure au moment du test.

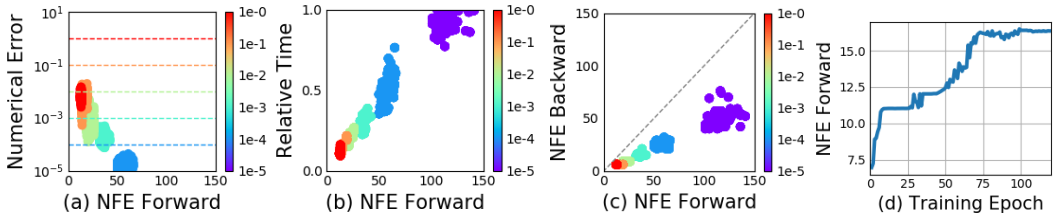


Figure 3: Statistiques d'un ODE-Net formé. (NFE = nombre d'évaluations de fonctions.)

Chiffre 3 c) montre un résultat surprenant: le nombre d'évaluations dans la passe arrière est à peu près la moitié de la passe avant. Cela suggère que la méthode de sensibilité adjointe est non seulement plus ef fi cace en mémoire, mais aussi plus ef fi cace en calcul que la rétropropagation directe via l'intégrateur, parce que cette dernière approche aura besoin de revenir en arrière à travers chaque évaluation de fonction dans la passe avant.

Profondeur du réseau Il n'est pas clair comment définir la «profondeur» d'une solution ODE. Une quantité associée est le nombre d'évaluations de la dynamique d'état caché requises, un détail délégué au solveur ODE et dépendant de l'état initial ou de l'entrée. Chiffre 3 d montre que le nombre d'évaluations de fonctions augmente tout au long de la formation, s'adaptant vraisemblablement à la complexité croissante du modèle.

4 flux de normalisation continus

L'équation discrétisée (1) apparaît également dans les flux de normalisation (Rezende et Mohamed , 2015) et le framework NICE (Dinh et coll. , 2014). Ces méthodes utilisent le théorème de changement de variables pour calculer les changements exacts de probabilité si les échantillons sont transformés via une fonction bijective F:

$$z_1 = F(z_0) \Rightarrow \text{Journal p}(z_1) = \text{Journal p}(z_0) - \text{Journal} \det \frac{\partial f}{\partial z_0} \tag{6}$$

Un exemple est le fl ux de normalisation planaire (Rezende et Mohamed , 2015):

$$z(t+1) = z(t) + euh(w^T z(t) + b), \quad \text{Journal p}(z(t+1)) = \text{journal p}(z(t)) - \text{Journal} \left| 1 + u^T \frac{\partial h}{\partial z} \right| \tag{7}$$

En règle générale, le principal goulot d'étranglement à l'utilisation de la formule de changement de variables est le calcul du déterminant du Jacobien $\partial f / \partial z$, qui a un coût cubique dans la dimension de z, ou le nombre d'unités cachées. Des travaux récents explorent le compromis entre l'expressivité de la normalisation des couches de flux et le coût de calcul (Kingma et coll. , 2016 ; Tomczak et Welling , 2016 ; Berg et coll. , 2018).

De manière surprenante, passer d'un ensemble discret de couches à une transformation continue simplifie le calcul du changement de la constante de normalisation:

Théorème 1 (Changement instantané des variables). Laisser z (t) être une variable aléatoire continue finie avec probabilité p (z (t)) dépend du temps. Laisser $\frac{dz}{dt} = F(z(t), t)$ être une équation différentielle décrivant une transformation continue dans le temps de z (t). En admettant que F est uniformément Lipschitz continue dans z et continue dans t, alors le changement de probabilité log suit également une équation différentielle,

$$\frac{\partial \text{Journal p}(z(t))}{\partial t} = -\text{tr} \frac{df}{dz(t)} \tag{8}$$

Preuve en annexe UNE . Au lieu du déterminant du log dans (6), nous n'avons plus besoin que d'une opération de trace. Contrairement aux fl ux finis standard, l'équation différentielle F n'a pas besoin d'être bijective, car si l'unicité est satisfaite, alors la transformation entière est automatiquement bijective.

A titre d'exemple d'application du changement instantané de variables, nous pouvons examiner l'analogue continu de l'écoulement planaire, et son changement de constante de normalisation:

$$\frac{\text{ré z (t)}}{\text{dt}} = \text{euh (w}_{\text{T}}\,\text{z (t) + b)}, \qquad \frac{\partial \text{ Journal p (z (t))}}{\partial \text{t}} = \text{- u}_{\text{T}}\,\frac{\partial \text{h}}{\partial \text{ z (t)}} \tag{9}$$

Étant donné une distribution initiale p (z (0)), nous pouvons échantillonner p (z (t)) et évaluer sa densité en résolvant cette ODE combinée

Utilisation de plusieurs unités cachées avec un coût linéaire Pendant que det n'est pas une fonction linéaire, la fonction trace is, ce qui implique tr ($_{\text{n}}$J $_{\text{n)}}$ = $_{\text{n}}$tr (J $_{\text{n)}}$. Ainsi, si notre dynamique est donnée par une somme de fonctions, alors l'équation différentielle de la densité logarithmique est également une somme:

$$\frac{\text{ré z (t)}}{\text{dt}} = \sum_{\text{n = 1}}^{\text{M}} \text{F}_{\text{n (z (t))}}, \qquad \frac{\text{ré Journal p (z (t))}}{\text{dt}} = \sum_{\text{n = 1}}^{\text{M}} \text{tr}\,\frac{\partial \text{f}_{\text{n}}}{\partial \text{ z}} \tag{dix}$$

Cela signifie que nous pouvons évaluer à moindre coût des modèles de flux ayant de nombreuses unités cachées, avec un coût uniquement linéaire dans le nombre d'unités cachées M. Évaluation de telles couches de flux `` larges '' en utilisant les coûts de normalisation des flux standard O (M $_{\text{3),}}$ ce qui signifie que les architectures NF standard utilisent plusieurs couches d'une seule unité cachée.

Dynamique dépendante du temps On peut spécifier les paramètres d'un fl ux en fonction de t, faire l'équation différentielle F(z (t), t) Echanger avec t. C'est le paramétrage est une sorte d'hyperréseau (Ha et coll. , 2016). Nous introduisons également un mécanisme de gating pour chaque unité cachée, $\frac{\text{ré z}}{\text{dt}}$ = $_{\text{n}}$σ $_{\text{n (}}$t) f $_{\text{n (}}$z) où σ $_{\text{n (}}$t) ∈ (0, 1) est un réseau de neurones qui apprend quand la dynamique F $_{\text{n (}}$z) devrait être appliqué. Nous appelons ces modèles des fl ux de normalisation continus (CNF).

4.1 Expériences avec des flux de normalisation continus

Nous comparons d'abord les fl ux plans continus et discrets lors de l'apprentissage d'échantillons à partir d'une distribution connue. Nous montrons qu'un CNF planaire avec M les unités cachées peuvent être au moins aussi expressives qu'une NF plane avec K = M couches, et parfois beaucoup plus expressives.

Correspondance de densité Nous con fi gurons le CNF comme décrit ci-dessus, et nous nous entraînons pour 10 000 itérations en utilisant Adam (Kingma et Ba , 2014). En revanche, le NF est formé pour 500000 itérations à l'aide de RMSprop (Hinton et coll. , 2012), comme suggéré par Rezende et Mohamed (2015). Pour cette tâche, nous minimisons KL (q (X) p (X)) comme fonction de perte où q est le modèle de flux et la densité cible p (·) peuvent être évalués. Chiffre 4 montre que CNF réalise généralement des pertes plus faibles.

Formation au maximum de vraisemblance Une propriété utile des fl ux de normalisation en temps continu est que nous pouvons calculer la transformation inverse pour environ le même coût que la passe avant, ce qui ne peut être dit pour normaliser les fl ux. Cela nous permet d'entraîner le fl ux sur une tâche d'estimation de densité en effectuant

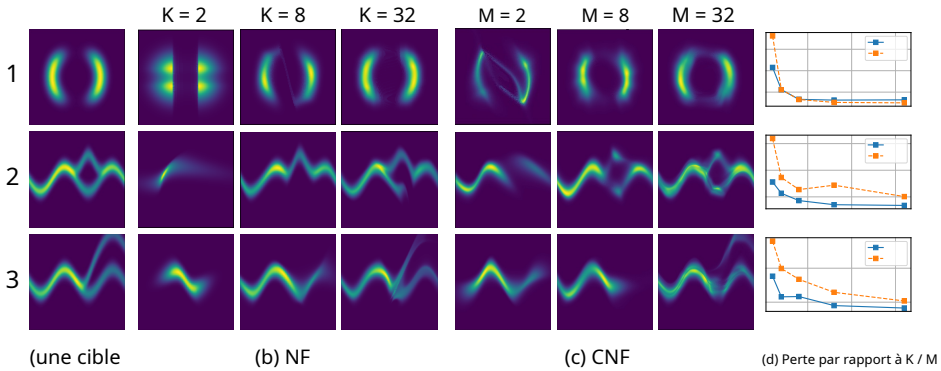
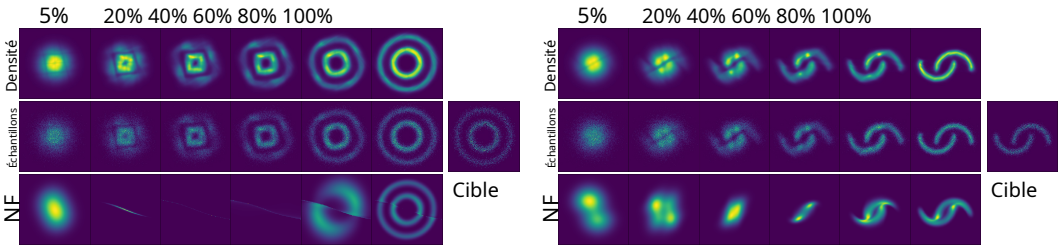


Figure 4: Comparaison des fl ux de normalisation par rapport aux fl ux de normalisation continus. La capacité du modèle des fl ux de normalisation est déterminée par leur profondeur (K), tandis que les fl ux de normalisation continus peuvent également augmenter la capacité en augmentant la largeur (M), ce qui les rend plus faciles à entraîner.



(a) Deux cercles (b) Deux lunes

Figure 5: Visualiser la transformation du bruit en données. Flux de normalisation en temps continu sont réversibles, nous pouvons donc nous entraîner à une tâche d'estimation de densité tout en étant capable d'échantillonner efficacement à partir de la densité apprise.

estimation du maximum de vraisemblance, qui maximise $E_{p(X)}[\text{Journal } q(X)]$ où $q(\cdot)$ est calculé en utilisant le théorème de changement approprié des variables, puis inverser le CNF pour générer échantillons de $q(X)$.

Pour cette tâche, nous utilisons 64 unités cachées pour CNF et 64 couches d'une unité cachée empilées pour NF. Chiffre 5 montre la dynamique apprise. Au lieu de montrer la distribution gaussienne initiale, nous affichons la distribution transformée après un petit laps de temps qui montre les emplacements des fl ux plans initiaux. Il est intéressant de noter que pour s'adapter à la distribution des deux cercles, le CNF fait pivoter les fl ux plans afin que les particules puissent être réparties uniformément en cercles. Bien que les transformations CNF soient lisses et interprétables, nous trouvons que les transformations NF sont très peu intuitives et que ce modèle a des difficultés à s'adapter à l'ensemble de données des deux lunes de la figure 5b .

5 Un modèle de séries chronologiques à fonction latente générative

L'application de réseaux neuronaux à des données échantillonnées de manière irrégulière telles que les dossiers médicaux, le trafic réseau ou les données de pointe neuronale est dif fi cile. Typiquement, les observations sont placées dans des bacs de durée fixe, et la dynamique latente est discrétisée de la même manière. Cela conduit à des dif fi cultés avec des données manquantes et des variables latentes dé fi nies. Les données manquantes peuvent être traitées à l'aide de modèles de séries chronologiques génératives (Álvarez et Lawrence , 2011 ; Futoma et coll. , 2017 ; Mei et Eisner , 2017 ; Soleimani et coll. , 2017a) ou l'imputation de données (Che et coll. , 2018). Une autre approche concatène les informations d'horodatage à l'entrée d'un RNN (Choi et coll. , 2016 ; Lipton et coll. , 2016 ; Du et coll. , 2016 ; Li , 2017).

Nous présentons une approche générative en temps continu pour la modélisation de séries chronologiques. Notre modèle représente chaque série chronologique par une trajectoire latente. Chaque trajectoire est déterminée à partir d'un état initial local, z_{t_0}, et un ensemble global de dynamiques latentes partagées dans toutes les séries chronologiques. Compte tenu des temps d'observation $t_0, t_1, \ldots, t_N$ et un état initial z_{t_0} , un solveur ODE produit $z_{t_1}, \ldots, z_{t_N}$, qui décrivent l'état latent à chaque Nous définissons formellement ce modèle génératif à travers une procédure d'échantillonnage:

$$z_{t_0} \sim p(z_{t_0}) \tag{11}$$

$$z_{t_1}, z_{t_2}, \ldots, z_{t_N} = \text{ODESolve}(z_{t_0}, f, \theta_f, t_0, \ldots, t_N) \tag{12}$$

$$\text{chaque } x_{t_i} \quad , \theta_x) \tag{13}$$

Une fonction F est une fonction invariante dans le temps qui prend la valeur z au pas de temps actuel et émet le pente: $\partial z(t)/\partial t = F(z(t), \theta_f)$. Nous paramétrons cette fonction à l'aide d'un réseau neuronal. Parce que F est invariant dans le temps, étant donné tout état latent z (t), toute la trajectoire latente est définie de manière unique. Extrapoler cette trajectoire latente nous permet de faire des prédictions arbitrairement loin en avant ou en arrière dans le temps.

Entraînement et prédiction Nous pouvons former ce modèle à variable latente comme un autoencodeur variationnel (Kingma et Welling , 2014 ; Rezende et coll. , 2014), avec des observations séquentielles. Notre réseau de reconnaissance est un RNN, qui consomme les données séquentiellement en arrière dans le temps, et hors met $q_\varphi(z_0 \mid x_1, x_2, \ldots, x_N)$. Un algorithme détaillé se trouve en annexe E . Utilisation des ODE comme le modèle génératif nous permet de faire des prédictions pour des points temporels arbitraires $t_1 \ldots t_M$ sur une chronologie continue.

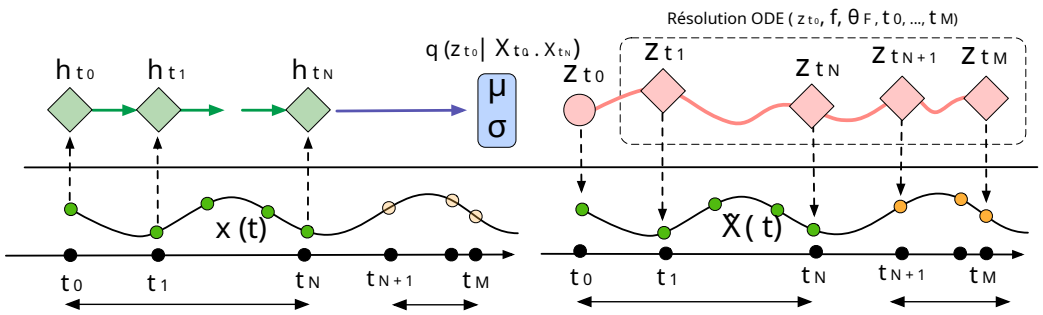


Figure 6: Graphique de calcul du modèle ODE latent.

Probabilités du processus de Poisson Le fait qu'une observation se soit produite nous dit souvent quelque chose sur l'état latent. Par exemple, un patient peut être plus susceptible de passer un test médical s'il est malade. Le taux d'événements peut être paramétré par une fonction de l'état latent: p (événement à la fois t | z (t)) = λ (z (t)).
Compte tenu de cette fonction de taux, la probabilité d'un ensemble d'indépendants temps d'observation de la dent dans l'intervalle [t démarrer, t finir] est donné par un processus de Poisson inhomogène (Paume , 1943): t finir

$$\text{Journal } p(t_1 \ldots t_N \mid t_{\text{démarrer}}, t_{\text{fin}}) = \sum_{i=1}^{N} \text{Journal } \lambda(z(t_{je})) - \int_{t_{\text{démarrer}}} \lambda(z(t))\, dt$$

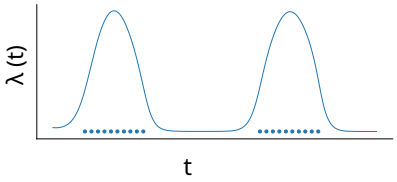


Figure 7: Ajustement d'un modèle dynamique ODE latent avec une vraisemblance de processus de Poisson. Les points indiquent les heures des événements. La ligne est l'intensité apprise λ (t) du processus de Poisson.

Nous pouvons paramétrer λ (·) en utilisant un autre réseau neuronal. Conveniently, nous pouvons évaluer à la fois la trajectoire latente et le
La vraisemblance du processus de Poisson en un seul appel à un solveur ODE. Chiffre 7 montre le taux d'événements appris par un tel modèle sur un jeu de données de jouet.

Une probabilité de processus de Poisson sur les temps d'observation peut être combinée avec une probabilité de données pour modéliser conjointement toutes les observations et les moments auxquels elles ont été faites.

5.1 Expériences ODE latentes de séries chronologiques

Nous étudions la capacité du modèle ODE latent d'ajuster et d'extrapoler des séries chronologiques. Le réseau de reconnaissance est un RNN avec 25 unités cachées. Nous utilisons un espace latent à 4 dimensions. Nous paramétrons la fonction dynamique F avec un réseau à une couche cachée avec 20 unités cachées. Le décodeur informatique p (X tj | z tj) est un autre réseau neuronal avec une couche cachée avec 20 unités cachées. Notre ligne de base était un réseau neuronal récurrent avec 25 unités cachées entraînées pour minimiser la probabilité logarithmique gaussienne négative. Nous avons formé une deuxième version de ce RNN dont les entrées ont été concaténées avec le décalage horaire jusqu'à l'observation suivante pour aider RNN avec des observations irrégulières.

Jeu de données spirale bidirectionnel Nous avons généré un ensemble de données de 1000 spirales bidimensionnelles, chacune commençant à un point différent, échantillonnées à 100 pas de temps équidistants. Le jeu de données contient deux types de spirales: la moitié dans le sens des aiguilles d'une montre et l'autre dans le sens inverse des aiguilles d'une montre. Pour faire la tâche plus réaliste, nous ajoutons du bruit gaussien aux observations.

Tableau 2: RMSE prédictif sur l'ensemble de test

# Observations	30/100	50/100	100/100
RNN	0,3937	0,3202	0,1813
ODE latent	0,1642	0,1502	0,1346

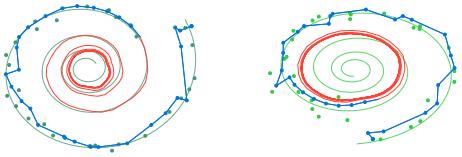
Série chronologique avec des points temporels irréguliers Pour générer des horodatages irréguliers, nous échantillonnons au hasard points de chaque trajectoire sans remplacement (n = { 30, 50, 100}). Nous rapportons l'erreur quadratique moyenne quadratique prédictive (RMSE) sur 100 points temporels s'étendant au-delà de ceux qui ont été utilisés pour la formation. Tableau 2 montre que l'ODE latente a un RMSE prédictif nettement plus faible.

Chiffre 8 montre des exemples de reconstructions en spirale avec 30 points sous-échantillonnés. Les reconstructions à partir de l'ODE latente ont été obtenues en échantillonnant à partir des trajectoires postérieures sur latentes et en les décodant

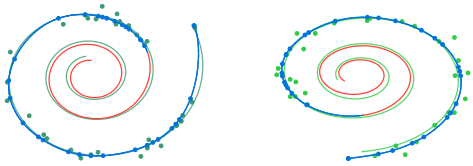
Figure 9: Trajectoires de l'espace de données décodées à partir d'une dimension variable de z t . La couleur indique progression dans le temps, commençant au violet et se terminant au rouge. Notez que le traj₀ Les secteurs à gauche sont dans le sens anti-horaire, tandis que les trajectoires à droite sont dans le sens des aiguilles d'une montre.

à l'espace de données. Des exemples avec un nombre variable de points temporels sont présentés en annexe F . Nous avons observé que les reconstructions et extrapolations sont cohérentes avec la vérité terrain quel que soit le nombre de points observés et malgré le bruit.

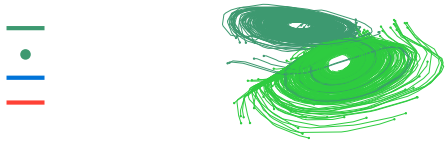
Interpolation d'espace latent Chiffre 8c spectacles trajectoires latentes projetées sur les deux premières dimensions de l'espace latent. Les trajectoires forment deux groupes séparés de trajectoires, l'un décodant en spirales dans le sens des aiguilles d'une montre, l'autre dans le sens inverse des aiguilles d'une montre. Chiffre 9 montre que les trajectoires latentes changent en douceur en fonction du point initial z (t₀), passage d'une spirale dans le sens des aiguilles d'une montre à une spirale dans le sens inverse des aiguilles d'une montre.



(a) Réseau neuronal récurrent



(b) Équation différentielle ordinaire neuronale latente



(c) Trajectoires latentes

Figure 8: (une): Reconstruction et extrapolation de spirales à points temporels irréguliers par un réseau neuronal récurrent. (b): Reconstructions et extrapolations par une ODE neurale latente. La courbe bleue montre la prédiction du modèle. Le rouge montre l'extrapolation. (c) Une projection de trajectoires d'EDO latentes à 4 dimensions inférées sur leurs deux premières dimensions. La couleur indique la direction de la trajectoire correspondante. Le modèle a appris une dynamique latente qui distingue les deux directions.

6 Portée et limites

Minibatching L'utilisation de mini-lots est moins simple que pour les réseaux neuronaux standard. On peut toujours regrouper les évaluations via le solveur ODE en concaténant les états de chaque élément de lot ensemble, créant ainsi un ODE combiné avec dimension ré × K. Dans certains cas, le contrôle des erreurs sur tous les éléments du lot peut nécessiter l'évaluation du système combiné. K fois plus souvent que si chaque système était résolu individuellement. Cependant, dans la pratique, le nombre d'évaluations n'a pas augmenté de façon substantielle lors de l'utilisation de minibatchs.

Unicité Quand la dynamique continue a-t-elle une solution unique? L'existence de Picard the- orem (Coddington et Levinson , 1955) indique que la solution à un problème de valeur initiale existe et est unique si l'équation différentielle est uniformément continue de Lipschitz en z et continue dans t. Ce théorème est valable pour notre modèle si le réseau de neurones a des poids fi nis et utilise des non-linéarités de Lipshitz, telles que ou alors .

Définition des tolérances Notre cadre permet à l'utilisateur de troquer la vitesse pour la précision, mais oblige l'utilisateur à choisir une tolérance d'erreur sur les passes avant et arrière pendant l'entraînement. Pour la modélisation de séquence, la valeur par défaut de a été utilisé. Dans les expériences de classification et d'estimation de la densité, nous avons pu réduire la tolérance à et , respectivement, sans dégrader les performances.

Reconstruire les trajectoires vers l'avant Reconstruire la trajectoire de l'état en exécutant la dynamique à l'envers peut introduire une erreur numérique supplémentaire si la trajectoire reconstruite diverge de l'original. Ce problème peut être résolu par le point de contrôle: stockage des valeurs intermédiaires de z sur le

passe avant, et reconstruire la trajectoire avant exacte en réintégrant à partir de ces points. Nous n'avons pas trouvé que c'était un problème pratique et nous avons vérifié de manière informelle que l'inversion de nombreuses couches de flux de normalisation continus avec des tolérances par défaut récupérait les états initiaux.

7 Travaux connexes

L'utilisation de la méthode adjointe pour la formation des réseaux de neurones en temps continu a été précédemment proposée (LeCun et coll. , 1988 ; Pearlmutter , 1995), mais n'a pas été démontré pratiquement. L'interprétation des réseaux résiduels He et coll. (2016a) en tant que solveurs ODE approximatifs ont stimulé la recherche sur l'exploitation de la réversibilité et du calcul approximatif dans les ResNets (Chang et coll. , 2017 ; Lu et coll. , 2017). Nous démontrons ces mêmes propriétés de manière plus générale en utilisant directement un solveur ODE.

Calcul adaptatif On peut adapter le temps de calcul en apprenant aux réseaux de neurones secondaires à choisir le nombre d'évaluations de réseaux récurrents ou résiduels (Tombes , 2016 ; Jernite et coll. , 2016 ; Figurnov et coll. , 2017 ; Chang et coll. , 2018). Cependant, cela introduit une surcharge à la fois au moment de la formation et du test, ainsi que des paramètres supplémentaires qui doivent être adaptés. En revanche, les solveurs ODE offrent des règles bien étudiées, peu coûteuses en calcul et généralisables pour adapter la quantité de calcul.

Backprop à mémoire constante grâce à la réversibilité Des travaux récents ont développé des versions réversibles de réseaux résiduels (Gomez et coll. , 2017 ; Haber et Ruthotto , 2017 ; Chang et coll. , 2017), ce qui donne le même avantage de mémoire constante que notre approche. Cependant, ces méthodes nécessitent des architectures restreintes, qui partitionnent les unités cachées. Notre approche n'a pas ces restrictions.

Apprendre les équations différentielles De nombreux travaux récents ont proposé d'apprendre des équations différentielles à partir de données. On peut entraîner des réseaux de neurones à réaction directe ou récurrente pour se rapprocher d'une équation différentielle (Raissi et Karniadakis , 2018 ; Raissi et coll. , 2018a ; Long et coll. , 2017), avec des applications telles que la simulation de fluides (Wiewel et coll. , 2018). Il y a également un travail important sur la connexion des processus gaussiens (GP) et des solveurs ODE (Schober et coll. , 2014). Les généralistes ont été adaptés aux équations différentielles (Raissi et coll. , 2018b) et peut naturellement modéliser les effets et les interventions en temps continu (Soleimani et coll. , 2017b ; Schulam et Saria , 2017). Ryder et coll. (2018) utilisent l'inférence variationnelle stochastique pour récupérer la solution d'une équation différentielle stochastique donnée.

Différenciation grâce aux solveurs ODE le bibliothèque (Farrell et coll. , 2013) implémente le calcul adjoint pour les solutions générales ODE et PDE, mais uniquement en rétropropagant à travers les opérations individuelles du solveur direct. La bibliothèque Stan (Carpenter et coll. , 2015) implémente l'estimation de gradient par des solutions ODE en utilisant l'analyse de sensibilité directe. Cependant, l'analyse de sensibilité directe est en temps quadratique dans le nombre de variables, tandis que l'analyse de sensibilité adjointe est linéaire (Carpenter et coll. , 2015 ; Zhang et Sandu , 2014). Melicher et coll. (2017) a utilisé la méthode adjointe pour former des modèles dynamiques latents sur mesure.

En revanche, en fournissant un produit vecteur-jacobien générique, nous permettons à un solveur ODE d'être formé de bout en bout avec tout autre composant de modèle différentiable. Alors que l'utilisation de produits vector-jacobiens pour résoudre la méthode adjointe a été explorée en contrôle optimal (Andersson , 2013 ; Andersson et coll. , Sous presse, 2018), nous mettons en évidence le potentiel d'une intégration générale des solveurs ODE boîte noire dans la différenciation automatique (Baydin et coll. , 2018) pour l'apprentissage en profondeur et la modélisation générative.

8 Conclusion

Nous avons étudié l'utilisation de solveurs ODE boîte noire comme composant de modèle, en développant de nouveaux modèles pour la modélisation de séries chronologiques, l'apprentissage supervisé et l'estimation de la densité. Ces modèles sont évalués de manière adaptative et permettent un contrôle explicite du compromis entre la vitesse de calcul et la précision. Enfin, nous avons dérivé une version instantanée de la formule de changement de variables et développé des fl ux de normalisation en temps continu, qui peuvent évoluer vers de grandes tailles de couche.

9 Remerciements

Nous remercions Wenyi Wang et Geoff Roeder pour l'aide avec les preuves, et Daniel Duckworth, Ethan Fetaya, Hossein Soleimani, Eldad Haber, Ken Caluwaerts et Daniel Flam-Shepherd pour leurs commentaires. Nous remercions Chris Rackauckas, Dougal Maclaurin et Matthew James Johnson pour des discussions utiles.

Les références

Mauricio A Álvarez et Neil D Lawrence. Sortie multiple convoluée efficace en termes de calcul Processus gaussiens. Journal of Machine Learning Research, 12 (mai): 1459-1500, 2011.

Brandon Amos et J Zico Kolter. OptNet: optimisation différentiable en tant que couche dans les réseaux de neurones. Dans Conférence internationale sur l'apprentissage automatique, pages 136 à 145, 2017.

Joel Andersson. Un cadre logiciel polyvalent pour l'optimisation dynamique. Thèse de doctorat, 2013.

Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings et Moritz Diehl. CasADi - A framework logiciel pour une optimisation non linéaire et un contrôle optimal. Calcul de programmation mathématique, Sous presse, 2018.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul et Jeffrey Mark Siskind. Différenciation automatique dans l'apprentissage automatique: une enquête. Journal de recherche sur l'apprentissage automatique, 18 (153): 1–153, 2018.

Rianne van den Berg, Leonard Hasenclever, Jakub M Tomczak et Max Welling. Sylvestre normalisation des fl ux pour l'inférence variationnelle. préimpression arXiv arXiv: 1803.05649, 2018.

Bob Carpenter, Matthew D Hoffman, Marcus Brubaker, Daniel Lee, Peter Li et Michael Betancourt. La bibliothèque mathématique Stan: différenciation automatique en mode inverse en c ++. préimpression arXiv arXiv: 1509.07164, 2015.

Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert et Elliot Holtham. Réversible architectures pour réseaux de neurones résiduels arbitrairement profonds. préimpression arXiv arXiv: 1709.03698, 2017.

Bo Chang, Lili Meng, Eldad Haber, Frederick Tung et David Begert. Réseaux résiduels multi-niveaux à partir de la vue des systèmes dynamiques. Dans Conférence internationale sur les représentations d'apprentissage, 2018. URL .

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag et Yan Liu. Neural récurrent réseaux pour les séries chronologiques multivariées avec des valeurs manquantes. Rapports scienti fi ques, 8 (1): 6085, 2018. URL .

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart et Jimeng Sun. Doctor AI: Prédire les événements cliniques via des réseaux neuronaux récurrents. Dans Actes de la 1ère conférence Machine Learning for Healthcare, volume 56 de Actes de recherche sur l'apprentissage automatique, pages 301–318. PMLR, 18-19 août 2016. URL .

Earl A Coddington et Norman Levinson. Théorie des équations différentielles ordinaires. Tata McGraw-Hill Education, 1955.

Laurent Dinh, David Krueger et Yoshua Bengio. NICE: composants indépendants non linéaires estimation. préimpression arXiv arXiv: 1410.8516, 2014.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez et Le Song. Processus ponctuels marqués récurrents: intégration de l'historique des événements dans le vecteur. Dans Conférence internationale sur la découverte des connaissances et l'exploration de données, pages 1555–1564. ACM, 2016.

Patrick Farrell, David Ham, Simon Funke et Marie Rognes. Dérivation automatisée de l'adjoint de programmes d'éléments fi nis transitoires de haut niveau. Journal du SIAM sur l'informatique scienti fi que, 2013.

Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov et Ruslan Salakhutdinov. Temps de calcul adaptatif spatial pour les réseaux résiduels. pré-impression arXiv, 2017.

J. Futoma, S. Hariharan et K. Heller. Apprendre à détecter la septicémie avec un classificateur RNN de processus gaussien multitâche. Impressions électroniques ArXiv, 2017.

Aidan N Gomez, Mengye Ren, Raquel Urtasun et Roger B Grosse. Le réseau résiduel réversible: Rétropropagation sans stocker les activations. Dans Progrès dans les systèmes de traitement de l'information neuronale, pages 2211 à 2221, 2017.

Alex Graves. Temps de calcul adaptatif pour les réseaux de neurones récurrents. préimpression arXiv arXiv: 1603.08983, 2016.

David Ha, Andrew Dai et Quoc V Le. Hypernetworks. préimpression arXiv arXiv: 1609.09106, 2016.

Eldad Haber et Lars Ruthotto. Architectures stables pour les réseaux de neurones profonds. Problèmes inverses, 34 (1): 014004, 2017.

E. Hairer, SP Nørsett et G. Wanner. Résolution d'équations différentielles ordinaires I - Problèmes non rigides. Springer, 1987.

Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. Apprentissage résiduel profond pour l'image reconnaissance. Dans Actes de la conférence IEEE sur la vision par ordinateur et la reconnaissance de formes, pages 770 à 778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. Mappages d'identité dans le résidu profond réseaux. Dans Conférence européenne sur la vision par ordinateur, pages 630–645. Springer, 2016b.

Geoffrey Hinton, Nitish Srivastava et Kevin Swersky. Conférence sur les réseaux de neurones pour l'apprentissage automatique 6a vue d'ensemble de la descente de gradient par mini-lots, 2012.

Yacine Jernite, Edouard Grave, Armand Joulin et Tomas Mikolov. Calcul des variables dans réseaux de neurones récurrents. préimpression arXiv arXiv: 1611.06188, 2016.

Diederik P Kingma et Jimmy Ba. Adam: Une méthode d'optimisation stochastique. préimpression arXiv arXiv: 1412.6980, 2014.

Diederik P. Kingma et Max Welling. Bayes variationnelles à codage automatique. Conférence internationale sur les représentations d'apprentissage, 2014.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever et Max Welling. Amélioration de l'inférence variationnelle avec un fl ux autorégressif inverse. Dans Progrès dans les systèmes de traitement de l'information neuronale, pages 4743 à 4751, 2016.

W. Kutta. Beitrag zur näherungsweisen Totalisateur d'intégration Differentialgleichungen. Zeitschrift für Mathematik und Physik, 46: 435–453, 1901.

Yann LeCun, D Touresky, G Hinton et T Sejnowski. Un cadre théorique pour la rétro-propagation. Dans Actes de l'école d'été des modèles connexionnistes de 1988, volume 1, pages 21–28. CMU, Pittsburgh, Pennsylvanie: Morgan Kaufmann, 1988.

Yann LeCun, Léon Bottou, Yoshua Bengio et Patrick Haffner. Apprentissage par gradient appliqué à reconnaissance de documents. Actes de l'IEEE, 86 (11): 2278-2324, 1998.

Yang Li. Représentation dépendante du temps pour la prédiction de séquences d'événements neuronaux. préimpression arXiv arXiv: 1708.00065, 2017.

Zachary C Lipton, David Kale et Randall Wetzel. Modélisation directe des données manquantes en séquences avec RNN: classi fi cation améliorée des séries chronologiques cliniques. Dans Actes de la 1ère conférence Machine Learning for Healthcare, volume 56 de Actes de recherche sur l'apprentissage automatique, pages 253–270. PMLR, 18-19 août 2016. URL .

Z. Long, Y. Lu, X. Ma et B. Dong. PDE-Net: apprentissage des PDE à partir de données. Impressions électroniques ArXiv, 2017.

Yiping Lu, Aoxiao Zhong, Quanzheng Li et Bin Dong. Au-delà des réseaux de neurones à couche fi nie: Relier les architectures profondes et les équations différentielles numériques. préimpression arXiv arXiv: 1710.10121, 2017.

Dougal Maclaurin, David Duvenaud et Ryan P Adams. Autograd: différenciation en mode inverse de Python natif. Dans Atelier ICML sur le Machine Learning automatique, 2015.

Hongyuan Mei et Jason M Eisner. Le processus neural de Hawkes: une auto-modulation neuronale processus ponctuel multivarié. Dans Progrès dans les systèmes de traitement de l'information neuronale, pages 6757–6767, 2017.

Valdemar Melicher, Tom Haber et Wim Vanroose. Dérivées rapides des fonctionnelles de vraisemblance pour Modèles basés sur ODE utilisant la méthode de l'état adjoint. Statistiques informatiques, 32 (4): 1621–1643, 2017.

Conny Palm. Intensitätsschwankungen im fernsprechverker. Ericsson Technics, 1943.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga et Adam Lerer. Différenciation automatique en pytorch. 2017.

Barak A Pearlmutter. Calculs de gradient pour les réseaux de neurones récurrents dynamiques: une enquête. IEEE Transactions sur les réseaux de neurones, 6 (5): 1212-1228, 1995.

Lev Semenovich Pontryagin, EF Mishchenko, VG Boltyanskii et RV Gamkrelidze. Les mathématiques théorie classique des processus optimaux. 1962.

M. Raissi et GE Karniadakis. Modèles physiques cachés: Apprentissage automatique des équations aux dérivées partielles non linéaires. Journal de physique computationnelle, pages 125 à 141, 2018.

Maziar Raissi, Paris Perdikaris et George Em Karniadakis. Réseaux de neurones à plusieurs étapes pour les données découverte guidée de systèmes dynamiques non linéaires. préimpression arXiv arXiv: 1801.01236, 2018a.

Maziar Raissi, Paris Perdikaris et George Em Karniadakis. Processus numériques gaussiens pour équations aux dérivées partielles non linéaires et dépendant du temps. Journal du SIAM sur l'informatique scienti fi que, 40 (1): A172 à A198, 2018b.

Danilo J Rezende, Shakir Mohamed et DaanWierstra. Rétropropagation stochastique et approximative inférence dans les modèles génératifs profonds. Dans Actes de la 31e Conférence internationale sur l'apprentissage automatique, pages 1278-1286, 2014.

Danilo Jimenez Rezende et Shakir Mohamed. Inférence variationnelle avec des fl ux normalisants. arXiv préimpression arXiv: 1505.05770, 2015.

C. Runge. Über die numerische Au fl ösung von Differentialgleichungen. Mathematische Annalen, 46: 167-178, 1895.

Lars Ruthotto et Eldad Haber. Réseaux de neurones profonds motivés par des équations différentielles partielles. préimpression arXiv arXiv: 1804.04272, 2018.

T. Ryder, A. Golightly, AS McGough et D. Prangle. Inférence variationnelle en boîte noire pour les équations différentielles stochastiques. Impressions électroniques ArXiv, 2018.

Michael Schober, David Duvenaud et Philipp Hennig. Solveurs probabilistes ODE avec Runge-Kutta moyens. Dans Progrès des systèmes de traitement de l'information neuronale 25, 2014.

Peter Schulam et Suchi Saria. Raisonnement hypothétique avec des processus gaussiens contrefactuels. arXiv préimpression arXiv: 1703.10651, 2017.

Hossein Soleimani, James Hensman et Suchi Saria. Modèles de joints évolutifs pour une incertitude fiable prédiction d'événement consciente. Transactions IEEE sur l'analyse de modèles et l'intelligence artificielle, 2017a.

Hossein Soleimani, Adarsh Subbaswamy et Suchi Saria. Modèles de réponse au traitement pour les raisonnement terfactuel avec interventions en temps continu, à valeur continue. préimpression arXiv arXiv: 1704.02038, 2017b.

Jos Stam. Fluides stables. Dans Actes de la 26e conférence annuelle sur l'infographie et techniques interactives, pages 121 à 128. ACM Press / Addison-Wesley Publishing Co., 1999.

Paul Stapor, Fabian Froehlich et Jan Hasenauer. Optimisation et analyse d'incertitude de l'ODE modèles utilisant une analyse de sensibilité adjointe du second ordre. bioRxiv, page 272005, 2018.

Jakub M Tomczak et Max Welling. Amélioration des auto-encodeurs variationnels à l'aide de Householder fl ow. préimpression arXiv arXiv: 1611.09630, 2016.

Steffen Wiewel, Moritz Becher et Nils Thuerey. Physique de l'espace latent: vers l'apprentissage de la évolution temporelle des fl uides. préimpression arXiv arXiv: 1802.10123, 2018.

Hong Zhang et Adrian Sandu. Fatode: une bibliothèque pour l'intégration linéaire directe, adjointe et tangente des ODE. Journal du SIAM sur l'informatique scienti fi que, 36 (5): C504-C523, 2014.