Étude de l'article "Meta Networks"

Eléonore Bartenlian ELEONORE.BARTENLIAN@U-PSUD.FR
Ashley Hill ASHLEY.HILL@U-PSUD.FR
Erwann Martin ERWANN.MARTIN@U-PSUD.FR
Adrien Pavao ADRIEN.PAVAO@U-PSUD.FR

Université Paris Saclay

Abstract

Les réseaux neuronaux ont montré leur efficacité dans de nombreuses applications. Cependant, ils nécessitent de grandes quantités de données étiquetées, et la généralisation à de nouveaux concepts à partir de peu d'exemples reste une tâche difficile. L'article que nous allons étudier, *Meta Networks* (Tsendsuren Munkhdalai, 2017), présente une nouvelle méthode d'apprentissage qui permet à un réseau neuronal d'apprendre et de généraliser une nouvelle tâche à partir d'un unique exemple. Dans ce rapport, nous commencerons par résumer l'article et exposer les problématiques principales ainsi que les résultats obtenus par les auteurs, puis nous présenterons nos propres résultats après avoir reproduit leurs expériences.

1. Introduction

L'être humain est capable de réagir vite à de nombreuses situations ; ainsi, la façon dont nous apprenons est flexible, continuelle et rapide. Elle dépend également de ce à quoi nous avons été exposés dans le passé. Nous sommes capables d'adapter notre comportement au retour d'information, ou *feedback*, reçu dans une situation ou un environnement donné (Harlow, 1949).

En revanche, en Machine Learning, ces feedbacks ne sont exploités que lors de la phase d'entraînement. Ainsi, ces systèmes classiques ne peuvent pas passer à l'échelle dans des environnements ou pour des tâches dynamiques : les réseaux neuronaux, une fois entraînés, ont des connexions et des architectures qui ne peuvent plus s'adapter. De plus, il sont sujets à l'oubli, nécessitent une grande quantité de données, et de nouvelles données peuvent corrompre ce qui a été précédemment appris.

Le but des auteurs était donc de concevoir un modèle de réseau neuronal intelligent qui soit capable d'adapter son comportement en fonction des feedbacks, et ce à n'importe quel moment, y compris pendant la phase de test.

2. La méthode

2.1. Le Meta learning

Le méta apprentissage, ou méta learning, est un sous-domaine de l'apprentissage automatique où les algorithmes d'apprentissage sont appliqués à des *méta données*. Bien que le terme "méta apprentissage" n'ait actuellement pas d'interprétation standard, l'objectif principal est d'utiliser ces méta données pour comprendre comment l'apprentissage automatique peut devenir flexible dans la résolution de problèmes d'apprentissage, et de pouvoir ainsi améliorer les performances des algorithmes d'apprentissage existants. Le but est donc d'apprendre à apprendre.

Des travaux précédents sur le Meta Learning ont présenté le problème comme un apprentissage à deux niveaux : un apprentissage sur différentes tâches s'effectue à un "méta" niveau pendant qu'un modèle "de base" travaille sur chaque tâche individuelle (Mitchell, 1993; Vilalta & Drissi, 2002). Ainsi, le méta apprenant infère des connaissances générales utiles pour différentes tâches, et l'apprenant de base va permettre de généraliser rapidement sur de nouvelles tâches.

2.2. Le One-Shot learning

Afin de tester la capacité à généraliser de leur modèle, les auteurs se penchent sur un cas particulier du One-Shot learning supervisé.

Ce type d'apprentissage est un scénario proche de la vie réelle : on donne à un apprenant une séquence de tâches, où chaque tâche correspond à une classification multi-classes avec un seul exemple par classe. Le défi principal provient du fait que les classes et les concepts varient selon les tâches.

2.3. Meta Networks

MetaNet est constitué des deux composantes principales (l'apprenant de base et le méta apprenant) ainsi que d'une mémoire externe qui va aider à apprendre et généraliser rapidement.

L'apprenant de base analyse la tâche en entrée, puis va donner un feedback au méta apprenant sous forme de méta informations (ici, des gradients de perte), puis le méta apprenant va se paramétrer et paramétrer l'apprenant de base afin de reconnaître les nouveaux concepts en entrée.

Les poids d'apprentissages de MetaNet évoluent selon deux échelles de temps :

- Des **poids lents** standards (qui évoluent lentement) mis à jour avec un algorithme d'apprentissage,
- Des **poids rapides** au niveau de la tâche (qui sont mis à jour dans le cadre de chaque tâche), et des poids rapide au niveau des exemples (qui sont mis à jour pour chaque exemple spécifique). Ces poids rapide permet des changements de tâches efficaces : en effet, le réseau ne dispose que de peu d'exemples pour chaque tâche et doit donc vite s'adapter.

Pour intégrer les poids lents standards et les poids rapides dans un réseau neuronal, les auteurs proposent une nouvelle méthode de *layer augmentation*. Dans l'apprenant de base, une couche lente est étendue avec ses poids rapides correspondants. L'entrée d'une couche augmentée est transformée par les poids lents et rapides, puis passée par une fonction ReLU. Les deux vecteurs d'activation en sortie sont ensuite additionnés composant par composant.

On distingue également deux types de fonctions de perte : une *embedding loss* définie pour permettre au méta apprenant de généraliser et une perte principale (*main loss*) utilisée pour l'objectif de l'entrée du réseau.

Pour entraîner MetaNet, on forme une séquence de tâches (Vinyals, 2016) où chaque tâche consiste en un ensemble de support (un ensemble avec un seul exemple étiqueté par classe) $\{x'_i, y'_i\}_{i=1}^{N}$ et un ensemble d'entraînement $\{x_i, y_i\}_{i=1}^{L}$. Les classes sont les mêmes pour l'ensemble de support et celui d'entraînement mais changent selon les tâches.

Pour tester l'apprentissage one-shot, les auteurs prennent une autre séquence de tâches avec des classes qui n'ont pas été encore vues, et le modèle va classifier ces exemples à partir de l'ensemble de support.

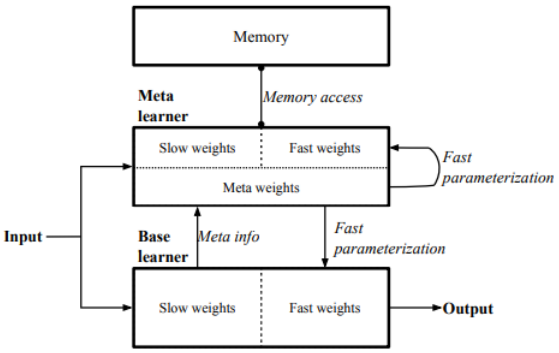

Figure 1. Architecture de Meta Networks.

2.4. L'algorithme

Prérequis

MetaNet a besoin d'un ensemble de support $\{x'_i, y'_i\}_{i=1}^{N}$ et d'un ensemble d'entraînement $\{x_i, y_i\}_{i=1}^{L}$. Il faut également l'apprenant de base b (un réseau neuronal), ainsi que le méta apprenant constitué de :

- Une fonction u d'apprentissage de la représentation dynamique,
- Des fonctions m et d qui génèrent les poids rapides au niveau des exemples et de la tâche.

En pratique, u, m et d sont des réseaux neuronaux.

Enfin, on a besoin des poids lents $\theta = (W, Q, Z, G)$, et d'un schéma d'augmentation de couches pour intégrer les poids lents et rapides.

Algorithme

On commence par prélever T exemples de l'ensemble de support. Pour chaque exemple, on calcule la perte $loss_{emb}$ afin d'obtenir les gradients (les méta informations). C'est fait grâce au réseau u du méta apprenant,

paramétrisé par les poids lents Q et les poids rapides au niveau de chaque tâche Q^*. Les poids $Q*$ sont générés de la manière suivante :

$$L_i = loss_{emb}(u(Q, x'_i), y'_i)$$
$$\nabla_i = \nabla_Q L_i$$
$$Q^* = d(G, \{\nabla\}_{i=1}^T)$$

Ensuite, c'est l'apprenant de base qui prend le relais. Il va estimer l'objectif principal grâce à la fonction de perte $loss_{task}$. b est paramétrisé par les poids lents W et les poids rapides au niveau des exemples W^*. Il va également utiliser l'ensemble de support afin de fournir au méta apprenant un feedback sur la nouvelle tâche. Les méta informations sont calculées de la manière suivante :

$$L_i = loss_{task}(b(W, x'_i), y'_i)$$
$$\nabla_i = \nabla_W L_i$$

Où L_i est la perte pour les exemples de support $\{x'_i, y'_i\}_{i=1}^N$, avec N le nombre d'exemples (dans le cas du one-shot, un seul par classe). ∇_i est le gradient de perte en fonction des paramètres W et représente nos méta informations.

Puis, le méta apprenant va prendre ce gradient et générer les poids rapides W^* grâce à la fonction m:

$$W_i^* = m(Z, \nabla_i)$$

Plus précisément, le réseau m de paramètres Z apprend le mapping entre le gradient de perte $\{\nabla_i\}_{i=1}^N$ (dérivé de l'apprenant de base b) et les poids rapides $\{W_i^*\}_{i=1}^N$.

Les poids rapides $\{W_i^*\}$ sont conservés à la ième position de la mémoire M, $M = \{W_i^*\}_{i=1}^N$.

Puis, des représentations dépendant des tâches $\{r'_i\}_{i=1}^N$ sont calculées grâce à u :

$$r'_i = u(Q, Q^*, x'_i)$$

où Q et Q^* sont intégrés en utilisant la méthode de layer augmentation.

Cela va nous permettre d'indexer la mémoire M avec la mémoire index R : $R = \{r'_i\}_{i=1}^N$. On va conserver r'_i à la ième position de R.

On va ensuite se pencher sur l'ensemble d'entraînement. Le méta apprenant va paramétrer l'apprenant de base avec les poids rapides W_i^* :

$$a_i = attention(R, r_i)$$
$$W_i^* = softmax(a_i)^T M$$

L'attention calcule la similarité entre la mémoire index et l'entrée. Enfin, on met à jour la perte d'entraînement pour chaque i :

$$L_{train} \leftarrow L_{train} + loss_{task}(b(W, W_i^*, x_i), y_i)$$

Et on peut mettre à jour les paramètres θ avec $\nabla_\theta L_{train}$.

3. Résultats de l'article

Les tests ont été effectués sur trois datasets : Omniglot (des caractères écrits à la main dans plus de 50 alphabets), Mini-ImageNet (100 types d'images) et MNIST (des chiffres manuscrits).

3.1. Tests de One-shot Learning

Le but de cette série de test était de voir à quel point le modèle est performant en one-shot learning.

3.1.1. Omniglot

Les auteurs ont fait des expériences sur Omniglot avec deux séparations test/train.

Dans un premier temps, ils ont réparti les données en 1200 classes d'entraînement et 423 classes de test. Ils ont également fait de la classification 5, 10, 15 et 20-way, c'est-à-dire que les ensembles de supports étaient constitués de 5, 10, 15 et 20 classes.

MetaNet améliore l'état de l'art de 0.5 à 2% de précision, et obtient entre 98.96% de précision (pour du 5-way) et 97% (pour du 20-way). La performance décroît lorsque le nombre de classes augmente, mais cela reste une chute faible (2%) comparé aux autres modèles qui eux chutent de 3 à 15%.

Dans un second test, les auteurs ont séparé les classes en 964 classes d'entraînement (30 alphabets) et 659 classes de test (20 alphabets). MetaNet dépasse légèrement les performances humaines mais pas l'approche probabiliste, même si les différences de performance sont faibles (moins de 1%). De plus, le modèle MetaNet ne dépend pas d'une connaissance extérieure sur la façon dont les lettres sont formées, contrairement à l'approche probabiliste.

Sur cette séparation des données, MetaNet est légèrement moins bon (98.45% pour du 5-way) ; en

effet, le nombre de classes en entraînement est moindre.

3.1.2. Mini-ImageNet

ImageNet est un jeu de données qui consiste à classifier 1000 différentes classes d'images naturelles. Sur du 5-way one-shot learning, MetaNet améliore l'état de l'art de 6% (49.21% de précision pour MetaNet à plus ou moins 0.96%).

3.2. Tests de généralisation

Le but de cette série de tests est de voir à quel point MetaNet est capable de généraliser.

3.2.1. Entraînement N-way et test K-way

Dans cette expérience, on teste si un modèle entraîné sur une tâche N-way one-shot est capable de généraliser à une autre tâche K-way avec $N \neq K$ sans s'entraîner sur la seconde tâche.

Le modèle est testé sur toutes les combinaisons du 5, 10, 15 et 20-way. MetaNet entraîné sur du 5-way obtient 93% de précision pour un test 20-way. Lorsqu'il est entraîné sur du 20-way et testé sur du 5-way, on obtient 99.55% de précision. Quand N est plus petit que K, c'est-à-dire lorsque le modèle est entraîné sur des taches plus faciles que les tâches de test, on observe une baisse de performances. Au contraire, et de façon assez naturelle, les modèles entraînés sur des tâches plus difficiles (N supérieur à K) et testés sur des tâches plus simples obtiennent des résultats meilleurs que pour $N < K$ ou $N = K$.

Malgré tout, les résultats restent bons dans tous les cas, ce qui suggère une grande flexibilité du modèle et donc une bonne généralisation des poids lents.

3.2.2. Paramétrisation rapide des poids fixes de l'apprenant de base

Dans ce test, les auteurs ont remplacé l'apprenant de base (on l'appellera apprenant cible) par un nouveau CNN au cours de l'évaluation. Les poids lents de ce réseau restaient fixes, tandis que les poids rapides sont paramétrés par le méta apprenant.

Les auteurs ont testé deux CNN de taille différentes en guise de nouvel apprenant de base. L'apprenant cible, optimisé progressivement, a de meilleures performances que les nouveaux CNN à poids fixes.

La différence de performance de ces modèles est grande dans les premières itérations, mais au fur et à mesure que le méta apprenant est exposé à des exemples one-shot, la précision des CNN converge. Cela montre que MetaNet parvient effectivement à apprendre à paramétrer un réseau neuronal avec des poids fixes.

3.2.3. Apprentissage en continu

MetaNet devrait être capable de retenir des informations passées, c'est à dire que la méta représentation devrait être indépendante du problème. Cette expérience vérifie si c'est bien le cas (lorsqu'on utilise le gradient de perte comme méta information).

Pour cela, les auteurs ont d'abord entraîné et testé le modèle sur Omniglot, puis ils ont continué l'entraînement sur MNIST. Au bout de quelques itérations sur MNIST, ils réévaluent le modèle sur le même ensemble Omniglot et comparent les performances. Une baisse de performance indique que le réseau est susceptible d'oublier ce qu'il a précédemment appris et que le modèle ne supporte pas l'apprentissage en continu. Au contraire, une augmentation des performances indique que MetaNet supporte l'apprentissage par transfert inverse, c'est-à-dire que le fait d'apprendre sur MNIST améliore les performances d'Omniglot.

Les améliorations des performances après avoir entraîné sur MNIST passent de 1.24% à -1.7% au cours de 7600 itérations. Les performances positives indiquent que l'entraînement sur le second problème améliore automatiquement les performances du premier problème. Cependant, cela n'est vrai que jusqu'à un certain point dans l'entraînement avec MNIST, point à partir duquel les poids commencent à oublier Omniglot. Après 2800 itérations, la précision baisse. Cependant, même après les 7600 itérations sur MNIST (données pour lesquelles la précision était de plus de 90%), la performance sur Omniglot ne descend que de 1.7%. Ces résultats montrent bien des propriétés d'apprentissage par transfert inverse.

4. Reproduction des expériences

4.1. Test de one-shot learning

Nous avons commencé par tester le one-shot learning sur Omniglot et sur MiniImageNet.

4.1.1. Omniglot

Nous avons suivi les préprocessings des auteurs, c'est-à-dire que nous avons redimentionné les images à une taille de 28x28. Puis, nous avons augmenté la base de donnée avec les images tournées de 90, 180 et 270 degrés.

Nous avons testé Omniglot sur du 5-way, en utilisant la découpe standard en train et test proposée par (Bren-

den M. Lake, 2015).

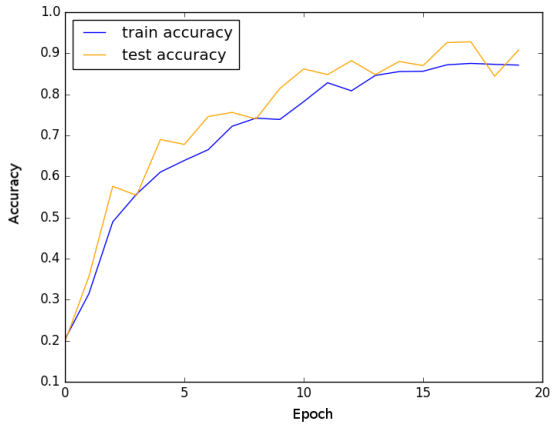

Figure 2. Précision sur 20 époques, Omniglot 5-way.

Faute de moyens matériels (pas de GPU et peu de puissance de calcul), nous n'avons pas pu faire tourner les calculs sur un grand nombre d'époques. On obtient au maximum 92% de précision sur 20 époques. C'est 6% de moins que les résultats de l'article, mais on peut supposer que les résultats auraient continué à augmenter si on avait pu laisser tourner le calcul plus longtemps.

Nous avons également testé du 10-way.

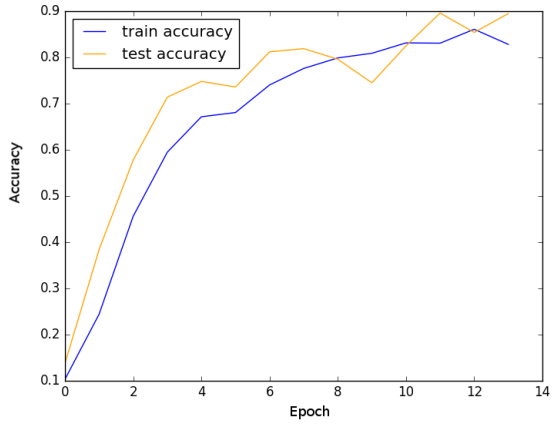

Figure 3. Précision sur 14 époques, Omniglot 10-way.

Les performances semblent évoluer de la même manière que pour du 5-way. On obtient 89% de précision après 14 époques. Il aurait fallu attendre que les modèles se stabilisent pour observer une réelle différence entre le 5-way et le 10-way. Néanmoins, on peut conclure qu'on obtient de bons résultats avec MetaNet.

4.1.2. MiniImageNet

La plupart des images utilisées par les auteurs ne sont plus accessibles actuellement. Il n'était donc pas possible de répliquer exactement leurs expériences sur MiniImageNet. La mise en place de ce test a donc été compliquée, à la fois à cause de ces images manquantes, mais également parce que ces données sont volumineuses et requièrent donc plus de ressources et de temps que les données d'Omniglot. Nous avons finalement réussi à faire tourner le modèle sur deux époques sans les images manquantes.

Epoque	Train loss	Train accuracy	Test loss	Test accuracy
0	519.51	0.1	120.71	0.2
1	345.38	0.1	120.71	0.2
2	345.39	0.1	120.71	0.2

Table 1. Résultats sur 3 époques, MiniImageNet 5-way.

On voit que le modèle commence à s'améliorer puisque la loss diminue, mais ces résultats ne nous permettent pas de conclure sur l'efficacité du modèle sur MiniImageNet. De plus, la perte sur l'ensemble de test est identique à chaque époque, ce qui indique que les poids du modèle ne sont peut-être pas correctement mis à jour.

4.2. Test de généralisation

Nous avons testé la généralisation de MetaNet en reproduisant l'expérience consistant à entraîner le modèle sur Omniglot puis sur MNIST.

4.2.1. Apprentissage en continu

Nous avons voulu vérifier les résultats obtenus lorsqu'on entraîne sur Omniglot, puis sur MNIST et qu'on teste régulièrement les performances sur Omniglot au cours de l'entraînement MNIST.

En effet, on peut se demander à quel point le modèle est capable à la fois de généraliser rapidement sur des nouvelles tâches tout en préservant les performances sur les tâches précédemment apprises.

Encore une fois, nous n'avons pas pu faire tourner les calculs autant qu'il aurait été nécessaire, mais nous pouvons tout de même nous faire une idée de l'efficacité du modèle avec les résultats obtenus. Nous avons entraîné Omniglot sur 10 époques, puis MNIST sur 10 époques également et testé la précision sur Om-

niglot.

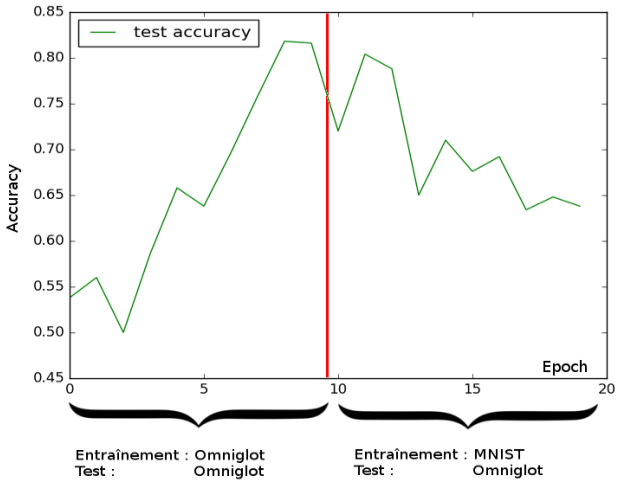

Figure 4. Précision sur Omniglot lorsque le modèle est entraîné sur Omniglot puis MNIST.

A la fin des 10 époques, on obtient une précision de 80% sur MNIST. On voit sur la *figure 4* que les performances sur Omniglot baissent rapidement, contrairement a ce qui a été montré dans l'article où les performances sur Omniglot continuaient à s'améliorer lorsque le modèle était entraîné sur MNIST, avant de baisser légèrement. En effet, la précision passe de plus de 80% après les 10 époques à 65% après les 10 époques de MNIST. Cela est probablement dû au fait que nous n'avons pas entraîné le modèle sur suffisamment d'époques avant de changer de base de données : le méta apprenant n'a pas eu le temps d'acquérir des connaissances solides sur la tâche.

5. Discussion

Nos résultats semblent, dans l'ensemble, en adéquation avec ceux obtenus dans l'article. Les différences proviennent surtout du fait que nous n'avons pas pu entraîner le modèle suffisamment longtemps.

On constate que la combinaison du one-shot learning et du méta learning permet d'éviter certains problèmes liés à la généralisation rapide avec peu de données et à l'apprentissage en continu des réseaux neuronaux. En effet, le modèle obtient d'excellents résultats sur les bases de données Omniglot et MiniImageNet en one-shot learning. Il est donc capable de généraliser à partir d'un seul exemple. De plus, il parvient à apprendre en continu et est beaucoup moins sujet à l'oubli que les réseaux classiques.

Une amélioration possible consisterait à trouver un nouveau type de méta information qui soit plus expressif et robuste que les gradients de perte, peut-être en s'inspirant du fonctionnement du cerveau. C'est un problème difficile car ces méta informations doivent être génériques, indépendantes du problème et suffisamment expressives.

On a vu que la paramétrisation rapide était une alternative efficace aux méthodes d'optimisation directes, mais l'intégration des poids lents et rapides peut poser problème. La solution qui a été trouvée est l'augmentation de couches. Cependant, cela devient difficile lorsqu'un réseau à beaucoup de paramètres. Dans l'idéal, il faudrait entraîner MetaNet de telle sorte qu'il soit capable de trouver son propre schéma d'augmentation de couches.

Enfin, MetaNet basé sur des réseaux récurrents pourrait être utile dans des taches de modélisation de séquences et de compréhension du langage.

6. Conclusion

Meta Networks semble être une méthode efficace mais peu intuitive et compliquée à mettre en oeuvre.

N'ayant pas le matériel nécessaire pour faire tourner des calculs coûteux, nous avons rencontré des difficultés et n'avons malheureusement pas pu reproduire toutes les expériences. En effet, MetaNet requière une puissance de calcul conséquente pour être entraîné en un temps raisonnable, et cela nous a limités. Nous n'avons notamment pas pu faire tourner le modèle sur un grand nombre d'époques, et n'avons pas non plus pu tester la classification 15 et 20-way.

Pour aller plus loin, il faudrait essayer d'appliquer MetaNet à d'autres jeux de données. En effet, l'article se focalise uniquement sur de la classification d'images, et on pourrait vouloir vérifier que les performances décrites ne dépendent pas d'un domaine d'application particulier.

References

Brenden M. Lake, Ruslan Salakhutdinov, Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

Harlow, Harry F. The formation of learning sets. *Psychological review, 56(1):51*, 1949.

Langley, P. Crafting papers on machine learning. In Langley, Pat (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML*

2000), pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Mitchell, Tom M, Thrun Sebastian B et al. Explanation-based neural network learning for robot control. *Advances in neural information processing systems*, 1993.

Tsendsuren Munkhdalai, Hong Yu. Meta networks. *ICML*, 2017. URL https://arxiv.org/pdf/1703.00837.pdf.

Vilalta, Ricardo and Drissi, Youssef. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 2002.

Vinyals, Oriol, Blundell Charles Lillicrap Tim Wierstra Daan et al. Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, 2016.