Desenvolvimento de um Método de *Unsupervised Domain Adaptation* para Domínios de Imagens

Renato Sérgio Lopes Júnior[1], **William Robson Schwartz**[1]

[1]Departamento de Ciência da Computação - Instituto de Ciências Exatas
Universidade Federal de Minas Gerais (UFMG)

{renato.junior,william}@dcc.ufmg.br

Abstract. *In this work, methods for solving the Unsupervised Homogeneous Domain Adaptation problem were developed. These methods were based on the concept of pseudo-labels and a multi-task learning architecture. The goal was to verify whether it is possible to peform the adaptation without resorting to complex metrics or domain confusion in an adversarial framework, like some proposed methods do.*

Resumo. *Neste trabalho foi realizado o desenvolvimento de métodos para solucionar o problema de Unsupervised Homogeneous Domain Adaptation que usavam o conceito de pseudo-labels e uma arquitetura de multi-task learning. O objetivo foi verificar se é possível realizar essa adaptação sem usar métricas complexas ou a ideia de confusão de domínio, como alguns métodos propostos na literatura fazem.*

1. Introdução

Técnicas de Aprendizagem de Máquina, do Inglês *Machine Learning* (ML), são cada vez mais utilizadas para resolução de problemas em inúmeras áreas [Weiss et al. 2016]. Um exemplo é o reconhecimento de pessoas e de objetos em imagens digitais.

Tradicionalmente, os algoritmos de ML usam um conjunto de dados para aprender a fazer predições para amostras que não estão nesse conjunto. A suposição feita é que esses dados usados durante a aprendizagem representam corretamente a distribuição original. Entretanto, caso essa suposição não seja válida, o modelo aprendido não apresentará bons resultados para as amostras de teste, que não pertencem ao conjunto de treinamento.

Isso acontece, por exemplo, quando há uma dificuldade em conseguir um grande conjunto de dados anotados para o treinamento que generalizem bem a real distribuição, visto que anotar dados é uma tarefa que consome muitos recursos, tanto humanos quanto financeiros. Assim, é desejado que exista uma forma de usar um conjunto que já esteja anotado para treinar o modelo, mesmo que haja uma discrepância entre os dados desse conjunto e os dados da aplicação alvo do modelo.

Para resolver esse problema, existem as técnicas de Adaptação de Domínio, do Inglês *Domain Adaptation*, que buscam contornar essa diferença entre as distribuições dos dados usados durante o treinamento e o teste, a fim de que o modelo aprendido tenha um bom resultado durante o teste.

Este trabalho teve como objetivo geral encontrar uma forma automática, ou que necessite do mínimo de supervisão, para realizar a adaptação de domínio com enfoque em Domínios de Imagens, de forma a facilitar esse procedimento em situações

onde os dados usados durante o treinamento não correspondem aos dados usados durante o teste. Mais especificamente, buscou-se uma forma de realizar a Adaptação de Domínio sem usar métricas complexas (como em [Long et al. 2015], [Long et al. 2016] e [Long et al. 2017]) ou a ideia de Confusão de Domínio em um *framework* de treinamento adversarial (como em [Ganin and Lempitsky 2015], [Tzeng et al. 2017] e [Kurmi and Namboodiri 2019]). O foco deste trabalho foi na tarefa de classificação de objetos, onde dada uma imagem de entrada, o modelo retorna a classe do objeto contido na imagem.

Neste artigo, são detalhados os métodos propostos e os resultados obtidos. Ele está organizado da seguinte forma: na seção 2 é dada uma definição mais formal do problema e uma visão geral dos principais métodos já propostos na literatura para resolvê-lo; na seção 3 são descritos os métodos propostos para realizar a adaptação; na seção 4 são apresentados os resultados dos experimentos realizados; por fim, na seção 5 é feita uma análise das atividades desenvolvidas no trabalho e dos possíveis próximos passos.

2. Revisão Bibliográfica

Um domínio é definido como uma combinação de um espaço de entrada $\mathcal{X}$, um espaço de saída $\mathcal{Y}$ e uma distribuição de probabilidade associada p, [Kouw and Loog 2018]. Dois domínios são diferentes quando há diferença de pelo menos um desses componentes entre eles.

As técnicas de *Domain Adaptation* são empregadas em situações onde há dois domínios diferentes, denominados *Source*, $\mathcal{S} = (\mathcal{X}_s, \mathcal{Y}_s, p_s)$, e *Target*, $\mathcal{T} = (\mathcal{X}_t, \mathcal{Y}_t, p_t)$, onde $\mathcal{X}_s, \mathcal{X}_t$ são os espaços de entrada, $\mathcal{Y}_s, \mathcal{Y}_t$ são os espaços de saída e p_s, p_t são as distribuições de probabilidade associada. O objetivo é treinar um modelo $\mathrm{y} = \theta(\mathbf{x})$ usando amostras anotadas de $\mathcal{S}$, $(\mathbf{x}_i^s, y_i^s)$, de forma a minimizar o risco em $\mathcal{T}$.

O problema pode ser categorizado de acordo com a diferença entre os domínios e a disponibilidade de dados anotados no domínio *Target*. O caso onde os espaços de entrada e de saída são os mesmos ($\mathcal{X}_t = \mathcal{X}_s$ e $\mathcal{Y}_t = \mathcal{Y}_s$) e apenas as distribuições de probabilidade p_s e p_t diferem entre os domínios, é denominado *Homogeneous Domain Adaptation*, ou Adaptação de Domínio Homogênea. Já o caso geral, onde os domínios $\mathcal{S}$ e $\mathcal{T}$ podem divergir de qualquer maneira, é denominado *Heterogeneous Domain Adaptation*, ou Adaptação de Domínio Heterogênea. Com relação à disponibilidade de dados anotados no domínio *Target*, o caso onde não há nenhuma amostra anotada do domínio *Target* é chamado de *Unsupervised Domain Adaptation*. Já quando há uma quantidade pequena de dados anotados do domínio *Target* disponível, é chamado de *Semi-supervised Domain Adaptation*. Neste trabalho, será considerado o caso de *Unsupervised Homogeneous Domain Adaptation*.

Na literatura, os métodos que obtém os melhores resultados para o caso *Unsupervised Homogeneous Domain Adaptation* são aqueles que usam métricas como a *Maximum Mean Discrepancy* (MMD) ou que usam um *framework* de treinamento adversarial que explora o conceito de *Domain Confusion*, ou confusão de domínio. Em [Long et al. 2015], são propostas as *Deep Adaptation Networks*. Os autores afirmam que, ao longo da rede neural, as *features* perdem a generalidade e passam a ficar cada vez mais específicas, perdendo sua transferibilidade. Assim, eles propõem a adição de um termo de regularização de adaptação multi-camada ao risco da rede, que leva em consideração

a métrica MK-MMD, *Multiple Kernel Maximum Mean Discrepancy*, que é a variante do MMD com múltiplos kernel. Essa métrica é definida como a distância entre os *mean embeddings* de duas distribuições p e q em um RKHS (*Reproducing Kernel Hilbert Space*). No artigo, os autores propõem essa arquitetura com base na AlexNet, sendo que as primeiras camadas convolucionais são congeladas, é feito *fine-tuning* das intermediárias e o termo de regularização baseado na MK-MMD é adicionado às últimas camadas *fully-connected*.

Uma outra forma de se realizar a adaptação é por meio de um *framework* adversarial. Os métodos com melhores resultados, como [Ganin and Lempitsky 2015], [Tzeng et al. 2017] e [Kurmi and Namboodiri 2019], usam um discriminador de domínio para explorar o conceito de Confusão de Domínio. Em [Tzeng et al. 2017], primeiramente uma rede neural convolucional (CNN) é treinada usando os dados anotados do domínio *Source*. Em seguida, uma CNN *encoder* para as amostras do domínio *Target* é aprendida usando o *encoder* já treinado anteriormente para o *Source* e um discriminador, que irá predizer qual é domínio de origem de cada *feature*. Após o treinamento adversarial, o *encoder* treinado para o domínio *Target* irá produzir *features* parecidas com as produzidas pelo *encoder* do domínio *Source*, o que possibilita o uso do classificador treinado anteriormente para classificar as amostras do *Target*, mesmo que ele tenha sido treinado usando apenas supervisão do *Source*.

Neste trabalho, buscou-se uma maneira de realizar a adaptação de domínio sem utilizar métricas complexas, como a MK-MMD, e sem utilizar um *framework* adversarial. Para tanto, foi explorada a ideia de *pseudo-labels*, que são anotações feitas de maneira automática para as amostras do domínio *Target*, visto que, no caso *Unsupervised*, não há nenhum dado anotado disponível de antemão. Alguns métodos já propostos usam *pseudo-labels* em conjunto com alguma métrica. Por exemplo, em [Wang et al. 2018], os autores propõem o *Manifold Embedded Distribution Alignment* (MEDA), que tem como ideia principal realizar a transformação das *features* para um espaço do *manifold* por meio da aprendizagem de um *manifold kernel* $\mathbb{G}$ e, em seguida, realizar o alinhamento dinâmico das distribuições. Como no caso *Unsupervised* não há anotações disponíveis do domínio *Target*, é utilizado um classificador base treinado no *Source* para obter *pseudo-labels* para o domínio *Target*.

3. Métodos Propostos

3.1. *Pseudo-labels*

O primeiro método proposto usa o conceito de *pseudo-labels* para realizar a adaptação. Tendo os dados anotados do domínio *Source* $\{(x_i^s, y_i^s)\}$ e as amostras não anotadas do domínio *Target* $\{x_i^t\}$, o objetivo é assinalar, de maneira iterativa, *labels* y_i^t para as amostras x_i^t. Assim, primeiramente o modelo é treinado usando $\{(x_i^s, y_i^s)\}$. Em seguida, a cada iteração, n elementos de $\{x_i^t\}$ são anotados com *pseudo-labels* y_i^t e é feito o *fine-tuning* do modelo usando as n amostras $\{(x_i^t, y_i^t)\}$. As n amostras são removidas de $\{x_i^t\}$ e se inicia uma nova iteração, onde serão anotadas outras n amostras. Para assinalar *pseudo-labels* às amostras, é feito o seguinte processo:

1. Contrói-se uma tabela com as *deep features* extraídas de uma das camadas da rede neural para todas as amostras do domínio *Source*;

2. Para cada amostra do domínio *Target*, x_i^t, é extraída a *deep feature* da mesma camada do passo anterior e se calcula uma métrica de distância para cada uma das linhas da tabela criada anteriormente. Com isso, obtém-se a distância relativa entre as representações latentes da amostra do domínio *Target* e de cada uma das amostras do domínio *Source*. A *label* y_i^t é definida como a *label* y_i^s da amostra do *Source* com a menor distância, obtendo-se o par (x_i^t, y_i^t);
3. São selecionados os n melhores pares (x_i^t, y_i^t) com base na distância (as n amostras com menor distância relativa).

O fluxograma apresentado na Figura 1 ilustra os passos deste método.

Uma outra versão desse método também foi implementada, onde são calculados os *centroids* de cada classe. Assim, ao se definir uma *label* para as amostras do domínio *Target*, é calculada a distância entre a *deep-feature* extraída e as representações dos *centroids*.

A suposição feita nesse método é que, por mais que exista uma divergência entre os domínios *Source* e *Target*, algumas amostras do *Target* terão uma representação latente, *deep feature*, que ficará próxima das representações das amostras da classe correspondente provenientes do domínio *Source*. Com isso, é possível assinalar *pseudo-labels* para algumas amostras (as que ficaram mais próximas, o que justifica selecionar as amostras que tiveram as menores distâncias). Além disso, o método parte do pressuposto que o *fine-tuning* do modelo via *backpropagation* usando essas amostras pseudo-anotadas é suficiente para corrigir a discrepância entre as distribuições, fazendo com que as *pseudo-labels* fiquem mais corretas ao longo das iterações.

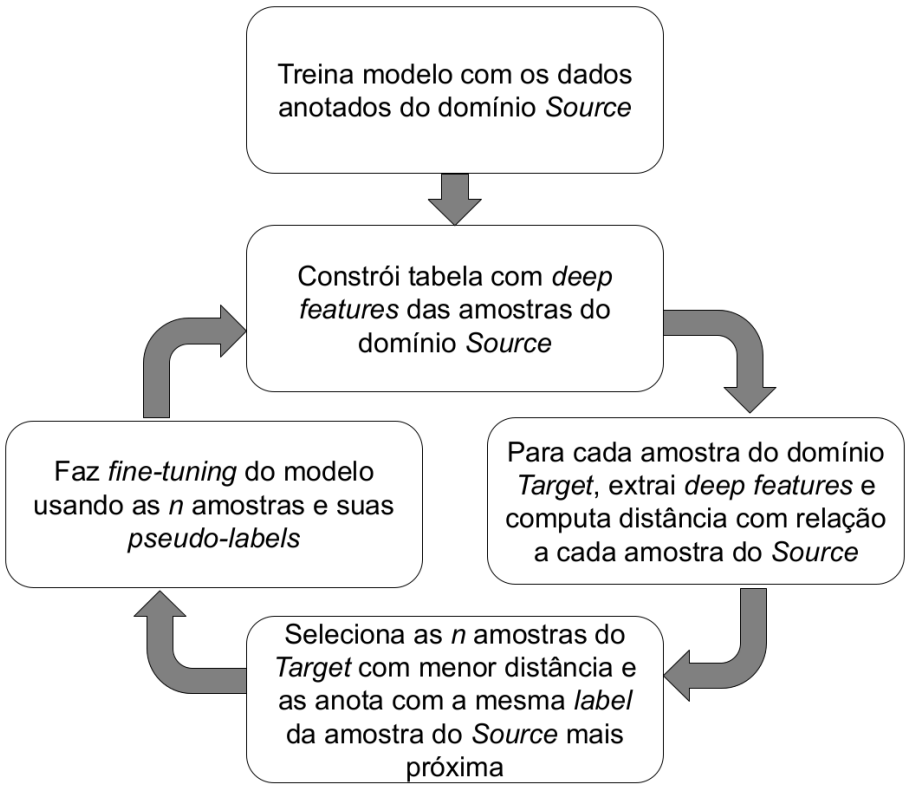

Figura 1. Fluxograma do método baseado em *Pseudo-labels*.

3.2. *Multi-task learning*

O segundo método proposto faz uso da aprendizagem multi-tarefa, ou *multi-task learning*. Além da tarefa de classificação entre as classes dos objetos, foi adicionada uma outra tarefa que classifica a imagem de entrada como sendo do domínio *Source* ou do domínio *Target* (isso é possível, pois, por mais que não se conheça a *label* das imagens do *Target*, sabe-se de qual domínio cada imagem é). A ideia por trás desse método foi verificar se realmente é necessário um *framework* de treinamento adversarial para que a rede neural não se especialize tanto no domínio *Source*, a fim de aumentar a sua transferibilidade para o domínio *Target*.

A primeira variação desse método treina a arquitetura *multi-task* usando os dados anotados do domínio *Source* $\{(x_i^s, y_i^s)\}$ e os dados não-anotados do domínio *Target* $\{(x_i^t)\}$. Para as amostras do domínio *Source*, são levadas em consideração tanto a perda (*loss*) de classificação dos objetos quanto a classificação dos domínios. Já para as amostras do domínio *Target*, é levada em consideração apenas a perda da classificação dos domínios.

Uma segunda variação tenta incorporar as *pseudo-labels* à arquitetura *multitask*: primeiramente, o modelo é treinado usando as amostras anotadas do domínio *Source*; em seguida, é feito o mesmo procedimento descrito na seção anterior, mas considerando as duas perdas: a de classificação dos objetos e a de classificação dos domínios.

A Figura 2 ilustra a arquitetura proposta.

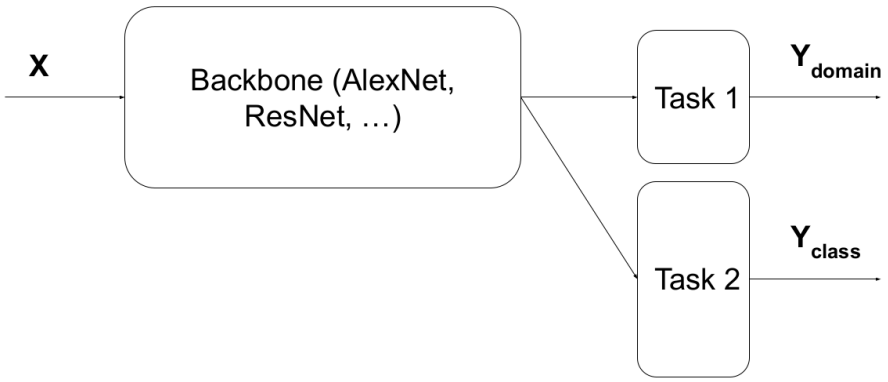

Figura 2. Arquitetura baseada em *multi-task learning*. Adiciona-se duas tarefas de classificação: a *Task* 1 irá classificar uma dada entrada X como sendo proveniente do domínio *Source* ou *Target*; a *Task* 2 irá classificar a entrada X em uma das classes do problema.

4. Experimentos

Foram conduzidos experimentos usando o *dataset* Office-31, [Saenko et al. 2010], que consiste de imagens de 31 diferentes objetos provenientes de 3 domínios: Amazon, imagens retiradas do catálago da loja *online* Amazon, que não possuem *background* e têm boas condições de iluminação; dSLR, imagens dos objetos retiradas com uma câmera digital, que possuem orientação e iluminação variadas e uma qualidade superior; e Webcam, imagens dos objetos retiradas com uma *webcam*, possuindo, assim, uma qualidade inferior. Exemplos de imagens desse *dataset* são apresentados na Figura 3.

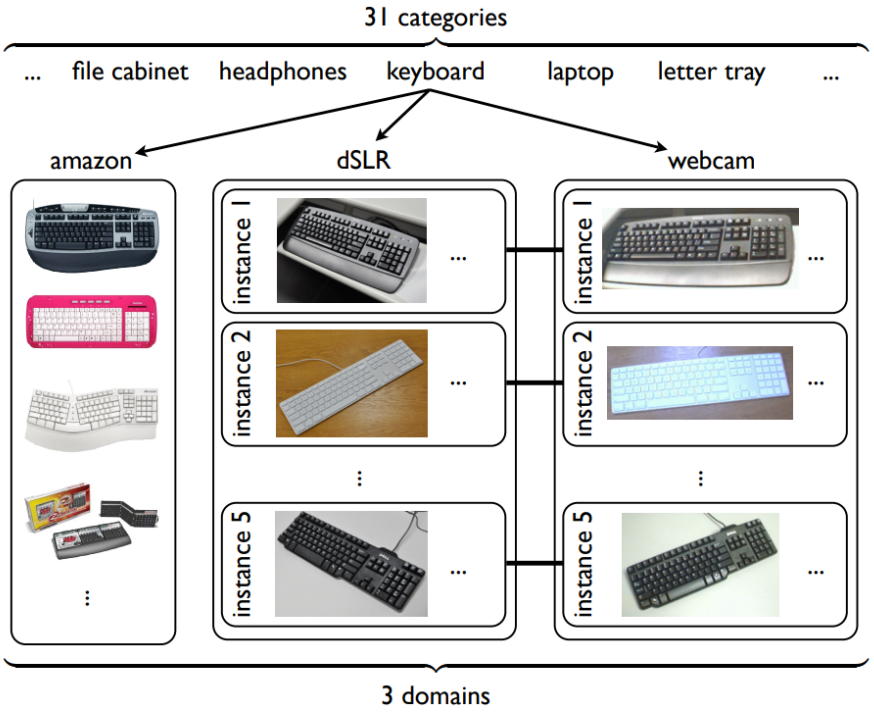

Figura 3. Exemplos dos domínios do dataset Office-31. Figura extraída de [Saenko et al. 2010].

Primeiramente, foi feito um experimento sem realizar qualquer adaptação. Nas Tabelas 1 e 2 são reportadas as acurácias no teste no domínio *Target* do modelo que foi treinado usando apenas dados anotados de um domínio *Source*, sem fazer qualquer tipo de adaptação. Nas Tabelas 1 e 2 são apresentados os resultados com a arquitetura AlexNet ([Krizhevsky et al. 2012] e ResNet-50 ([He et al. 2016]), respectivamente. Nota-se que, de fato, quanto maior é a diferença entre os domínios (no caso o domínio Amazon é o mais díspare), maior será o impacto na acurácia. Esses valores também mostram que, por mais que as redes neurais profundas tenham um grande poder de generalização, ainda assim, quando há uma grande diferença entre os dados usados durante o treinamentos e os dados de teste, o desempenho dessas redes sofre um grande impacto, tanto as arquiteturas mais antigas, como a AlexNet, quanto as mais recentes, como a ResNet-50.

Source/Target	Amazon	DSLR	Webcam
Amazon	$0.7413 \pm .0217$	$0.4438 \pm .0423$	$0.4456 \pm .0255$
DSLR	$0.2326 \pm .0326$	$0.8857 \pm .0291$	$0.7244 \pm .0738$
Webcam	$0.2636 \pm .0270$	$0.8447 \pm .0220$	$0.9193 \pm .0246$

Tabela 1. Acurácias no teste no domínio *Target* para diferentes combinações de Domínio *Source* e *Target* na arquitetura AlexNet. O modelo foi treinado usando apenas dados anotados do domínio *Source* (linhas), sem nenhum tipo de adaptação. Na diagonal principal, é retratado o caso tradicional de *Machine Learning*, onde os domínios *Source* e *Target* são iguais.

Em seguida, foram feitos testes com todas as combinações possíveis de pares de domínio *Source* e *Target*: A→D, A→W, D→A, D→W, W→A e W→D, onde A, D e W

Source/Target	Amazon	DSLR	Webcam
Amazon	$0.8188 \pm .0083$	$0.6303 \pm .0300$	$0.6245 \pm .0385$
DSLR	$0.4840 \pm .0068$	$0.9607 \pm .0135$	$0.8936 \pm .0162$
Webcam	$0.5359 \pm .0197$	$0.9732 \pm .0109$	$0.9766 \pm .0117$

Tabela 2. Acurácias no teste no domínio *Target* para diferentes combinações de Domínio *Source* e *Target* na arquitetura ResNet-50. O modelo foi treinado usando apenas dados anotados do domínio *Source* (linhas), sem nenhum tipo de adaptação. Na diagonal principal, é retratado o caso tradicional de *Machine Learning*, onde os domínios *Source* e *Target* são iguais.

são os domínios Amazon, dSLR e Webcam, respectivamente. Utilizou-se a arquitetura AlexNet [Krizhevsky et al. 2012], que consiste de 5 camadas convolucionais e 3 camadas *fully-connected*, visto que os outros métodos de *Domain Adaptation* também usam essa arquitetura.

Na Figura 4, são apresentados os resultados da execução do método baseado em *pseudo-labels*. Foram extraídas *deep-features* da segunda camada *fully-connected* e o parâmetro n, o número de amostras do *Target* que serão pseudo-anotadas a cada iteração, foi definido como 10. O método foi executado por 100 iterações (como alguns domínios possuíam menos de 1000 imagens, para alguns pares o método é executado por menos iterações). Nos gráficos, pode ser visualizada a evolução da acurácia no teste tanto no domínio *Source* quanto no domínio *Target*. Nota-se que esse método conseguiu apenas um ganho marginal na acurácia no domínio *Target* em todos os pares *Source/Target* e que, além disso, a partir uma iteração o desempenho no *Target* passa a diminuir (*Negative Transfer*), o que sugere que estão sendo definidas *pseudo-labels* erradas para as amostras do *Target*[1].

Na Figura 5, são apresentados os resultados do mesmo método baseado em *pseudo-labels*, mas utilizando um *centroid* das *deep-features* de cada classe do *Source* para se calcular a distância e definir a *pseudo-label* para as amostras do *Target*. Mais uma vez, houve apenas um ganho marginal na acurácia no teste para o domínio *Target*, sendo que, para alguns pares *Source/Target*, os resultados usando o *centroid* ficaram piores que o método original, sem usar *centroids*.

Os resultados obtidos usando a arquitetura *Multi-task* são apresentados na Tabela 3. O modelo foi treinado usando as amostras anotadas do domínio *Source* e as amostras não-anotadas do *Target*. Para as imagens do *Source* foram considerados os custos das duas tarefas. Já para as imagens do *Target* foi considerada apenas o custo da tarefa de classificação entre os domínios. Comparando esses resultados com os obtidos sem realizar nenhuma adaptação (Tabela 1), verifica-se que a adição dessa nova tarefa prejudicou bastante o desempenho do modelo, até mesmo quando os domínios *Source* e *Target* são os mesmos. Isso indica que apenas a tarefa de classificação entre os domínios (sem a adição de uma *Gradient Reversal Layer*), além de não melhorar a transferibilidade das *features*, causa um impacto negativo na tarefa de classificação.

[1]Provavelmente, o *fine-tuning* do modelo usando as amostras pseudo-anotadas não é suficiente para alinhar as distribuições p_s e p_t, fazendo com que, ao longo das iterações, cada vez mais amostras do *Target* sejam assinaladas com a classe errada

Source/Target	Amazon	DSLR	Webcam
Amazon	$0.4617 \pm .0436$	$0.3196 \pm .0533$	$0.2795 \pm .0337$
DSLR	$0.1160 \pm .0204$	$0.5393 \pm .0567$	$0.2526 \pm .0633$
Webcam	$0.1718 \pm .0377$	$0.3714 \pm .0598$	$0.5333 \pm .0593$

Tabela 3. Acurácias no teste no domínio *Target* para diferentes combinações de Domínio *Source* e *Target* na arquitetura *Multi-task*. Foi utilizada a arquitetura AlexNet como *backbone*, e o treinamento foi feito utilizando as amostras anotas do *Source* e as amostras não-anotadas do *Target*.

Na Figura 6, são apresentados os resultados do método de *pseudo-labels* usando a arquitetura *multi-task*. Os testes foram realizados definindo $n = 10$ e utlizando a AlexNet como *backbone*. Foram extraídas *deep-features* da segunda camada *fully-connected* do *backbone*, no caso a AlexNet. Nota-se que as acurácias iniciais são bem inferiores ao método usando a arquitetura original, sem *multi-task*, o que indica, como mencionado anteriormente, que a adição da tarefa de classificação entre os domínios não aumentou a transferibilidade do modelo, além de piorar a tarefa de classificação. Ao longo das iterações, é possível verificar o mesmo comportamento do método baseado em *pseudo-labels* usando a arquitetura sem *multi-task* (Figura 4): há apenas um aumento marginal na acurácia no teste no domínio *Target*.

Assim, analisando esses resultados, pode-se concluir que não houve um ganho significativo na acurácia no teste do domínio *Target*. Idealmente, haveria um grande aumento na acurácia no *Target*, enquanto que a acurácia no *Source* ficaria constante ou sofreria uma queda[2]. Todavia, os resultados mostram que em todos os métodos implementados só foi observado um aumento marginal nessa métrica.

Uma possível justificativa para esses resultados é que apenas a realização do *fine-tuning* do modelo usando as *pseudo-labels* das amostras do domínio *Target* não é suficiente para alinhar as distribuições, não aumentando, assim, o desempenho do modelo no domínio *Target*. Uma outra hipótese é que o método empregado para assinalar *pseudo-labels* às amostras não-anotadas do domínio *Target*, baseado em uma métrica de distância entre as *deep-features*, pode não ser o ideal, assinalando pseudo-anotações erradas às amostras.

Quanto aos resultados obtidos com a arquitetura *multi-task*, uma possível justificativa é que a adição da tarefa de classificação entre os domínios não auxilia no aumento da transferibilidade do modelo. Pelo contrário, ela aumenta a discrepância entre as *features*. Assim, esse discriminador de domínios deve ser utilizado com um *framework* adversarial, como é feito em [Tzeng et al. 2017] e em [Kurmi and Namboodiri 2019]. De fato, os métodos que melhor resolvem o problema de *Unsupervised Domain Adaptation* são os que exploram a ideia de Confusão de Domínio, por meio do uso de um discriminador de domínio treinado de forma adversarial.

[2]Como o objetivo é melhorar o desempenho do modelo no domínio *Target*, uma queda no desempenho no domínio *Source* é aceitável.

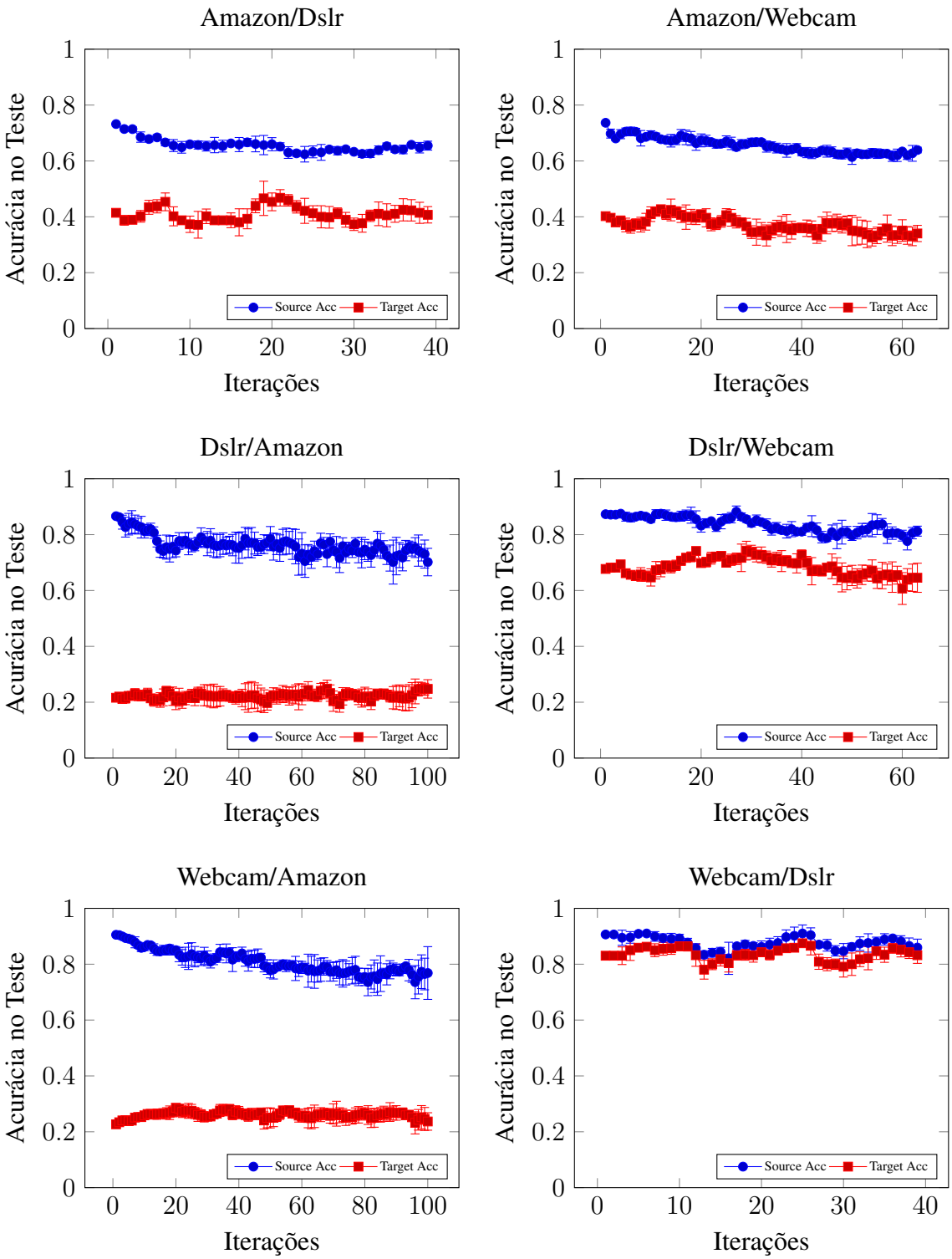

Figura 4. Resultados obtidos com o método baseado em Pseudo-labels. São reportadas as acurácias no teste para todos os pares de domínios *Source/Target* ao longo das iterações.

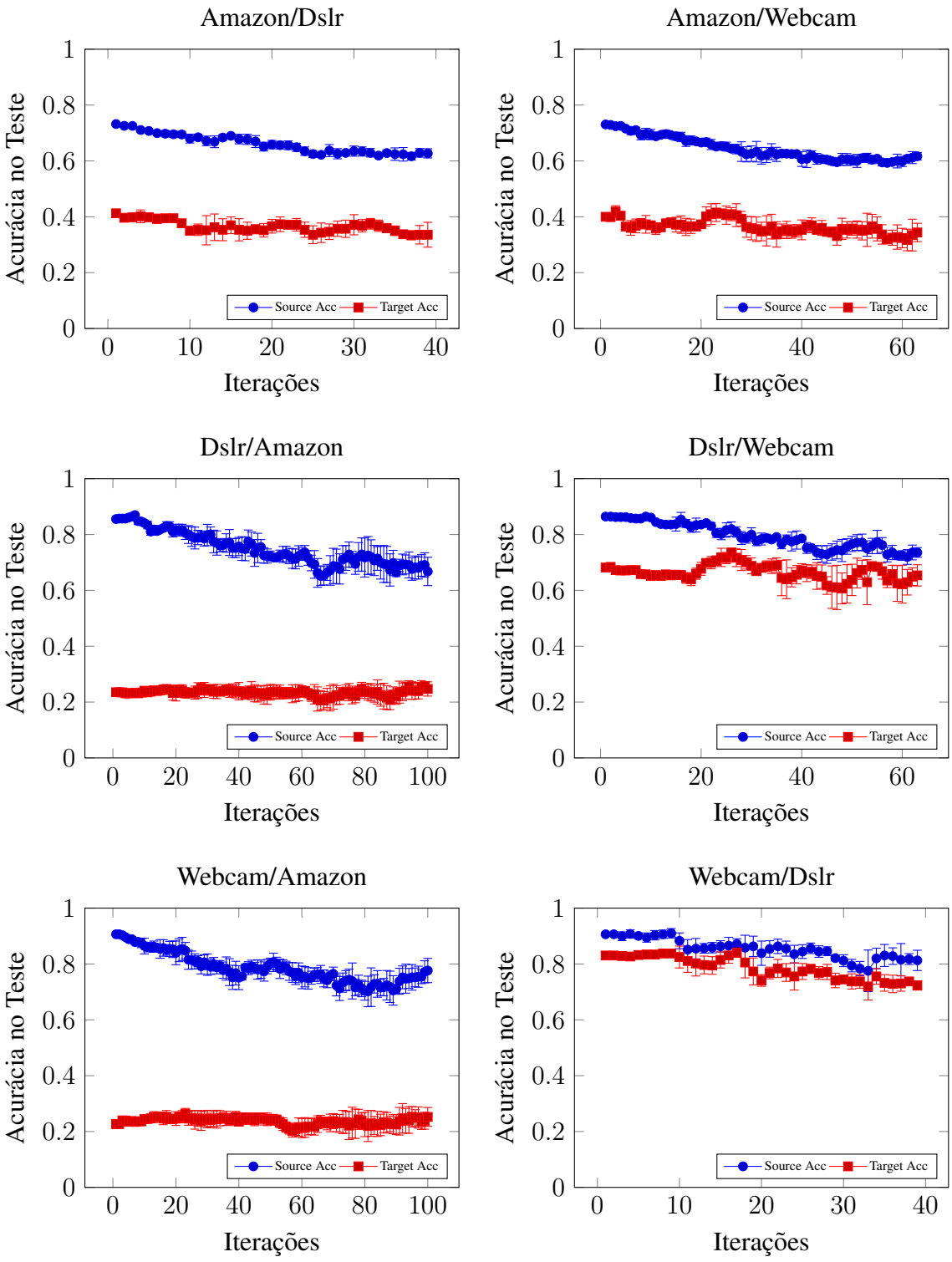

Figura 5. Resultados obtidos com o método baseado em Pseudo-labels usando o centroid da classe para assinalar as *pseudo-labels*. São reportadas as acurácias no teste para todos os pares de domínios *Source/Target* ao longo das iterações.

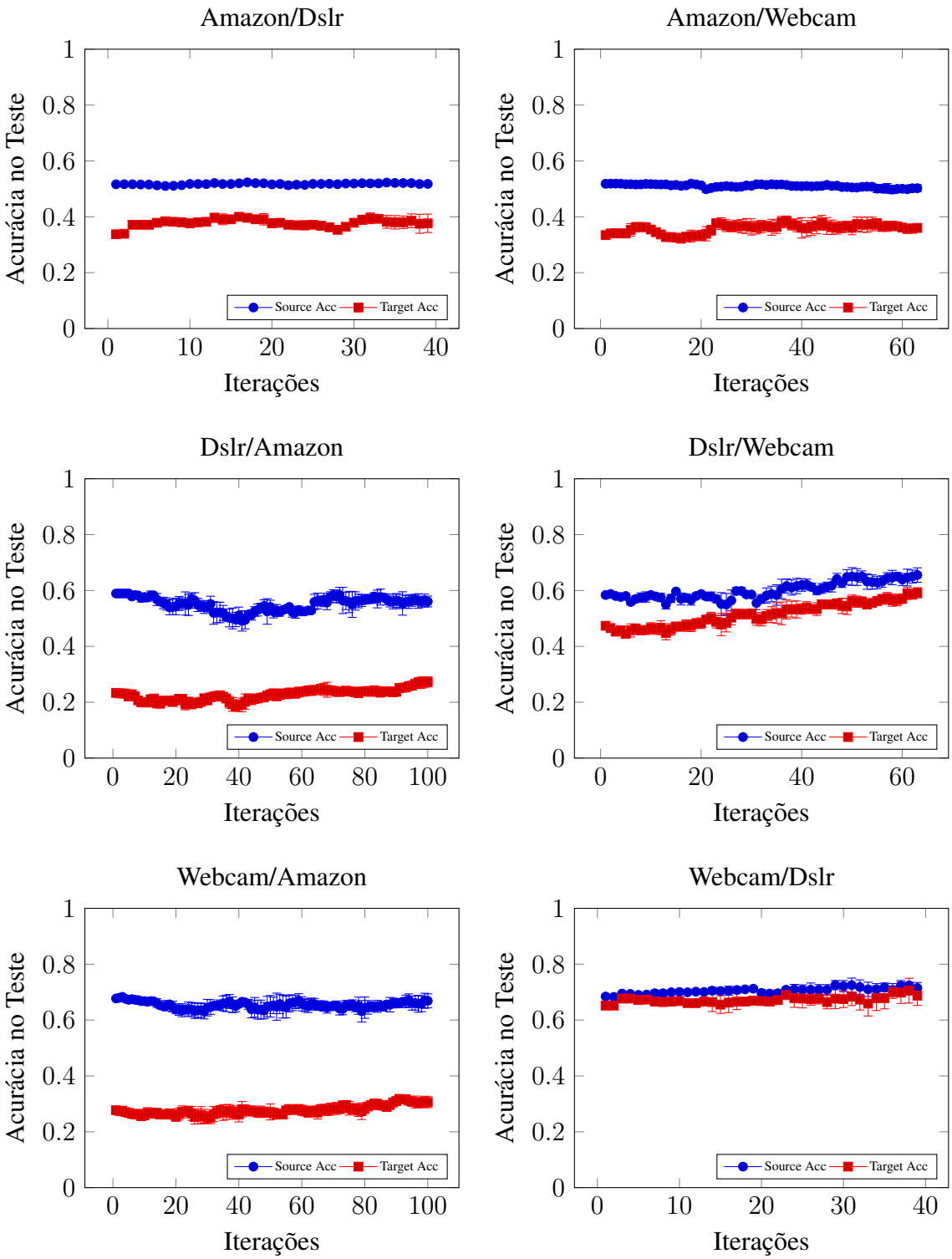

Figura 6. Resultados obtidos com o método baseado em *Pseudo-labels* utlizando a arquitetura *Multi-task*. São reportadas as acurácias no teste para todos os pares de domínios *Source/Target* ao longo das iterações.

5. Conclusão

Neste trabalho, foi realizada a pesquisa por um método de *Unsupervised Homogeneous Domain Adaptation* aplicado a domínios de imagens, com foco na tarefa de classificação. Este método possibilitaria treinar um modelo usando dados anotados de um domínio *Source* para ser usado em um domínio *Target*, mesmo que as distribuições dos dados desses domínios fossem diferentes. Buscou-se por uma forma de realizar a adaptação sem utilizar métricas complexas ([Long et al. 2015], [Long et al. 2016], [Long et al. 2017], [Wang et al. 2018]) ou um *framework* de treinamento adversarial ([Ganin and Lempitsky 2015], [Tzeng et al. 2017], [Kurmi and Namboodiri 2019]). Para tanto, foi desenvolvido um método usando o conceito de *pseudo-labels*, que anotava automaticamente as amostras do domínio *Target* e usava essas pseudo-anotações para fazer *fine-tuning* do modelo, a fim de corrigir a discrepância entre as distribuições. Também foi desenvolvida uma nova arquitetura baseada em *multi-task learning*, que contava com uma tarefa de classificação entre os domínios, além da tarefa de classificação dos objetos. Todavia, tanto o método baseado em *pseudo-labels* quanto o baseado em *multi-task learning* não obtiveram um ganho significativo na acurácia no teste no domínio *Target*, obtendo apenas um ganho marginal nessa métrica. Assim, possíveis próximos passos são: encontrar um novo método para assinalar as *pseudo-labels* às amostras do *Target*; tentar realizar a redução da dimensionalidade das *deep-features* extraídas, usando métodos como PCA e PLS; e, por fim, usar *deep-features* de mais de uma camada da rede neural, a fim de obter informações de diferentes níveis de abstração.

Referências

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1180–1189. JMLR.org.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Kouw, W. M. and Loog, M. (2018). An introduction to domain adaptation and transfer learning. *CoRR*, abs/1812.11806.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114.

Kurmi, V. K. and Namboodiri, V. P. (2019). Looking back at labels: A class based domain adaptation technique. In *International Joint Conference on Neural Networks (IJCNN)*.

Long, M., Cao, Y., Wang, J., and Jordan, M. I. (2015). Learning transferable features with deep adaptation networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 97–105. JMLR.org.

Long, M., Zhu, H., Wang, J., and Jordan, M. I. (2016). Unsupervised domain adaptation with residual transfer networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 136–144, USA. Curran Associates Inc.

Long, M., Zhu, H., Wang, J., and Jordan, M. I. (2017). Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2208–2217. JMLR.org.

Saenko, K., Kulis, B., Fritz, M., and Darrell, T. (2010). Adapting visual category models to new domains. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 213–226, Berlin, Heidelberg. Springer-Verlag.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, J., Feng, W., Chen, Y., Yu, H., Huang, M., and Yu, P. S. (2018). Visual domain adaptation with manifold embedded distribution alignment. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, pages 402–410, New York, NY, USA. ACM.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning. *Journal of Big Data*, 3(1):9.