Uma abordagem para coleta e análise de dados de configurações em redes neurais profundas *

Débora Pina[1], Liliane Kunstmann[1], Daniel de Oliveira[2], Patrick Valduriez[3], Marta Mattoso[1]

[1]Universidade Federal do Rio de Janeiro (PESC/COPPE/UFRJ), Brasil

{dbpina, lneves, marta}@cos.ufrj.br

[2]Universidade Federal Fluminense (IC/UFF), Brasil

danielcmo@ic.uff.br

[3]Inria, University of Montpellier, CNRS, LIRMM, França

Patrick.Valduriez@inria.fr

Resumo. *O tempo de duração do ciclo de vida no aprendizado por meio de redes neurais profundas depende do acerto em decisões de configuração de dados que levem ao sucesso na obtenção de modelos. A análise de hiperparâmetros e dados da evolução da rede permite adaptações que diminuem o tempo de duração do ciclo de vida. No entanto, há desafios não apenas na coleta de hiperparâmetros, mas também na modelagem dos relacionamentos entre esses dados. Este trabalho apresenta uma abordagem centrada em dados de proveniência para enfrentar esses desafios, propondo uma coleta com flexibilidade na escolha e representação de dados a serem analisados. Experimentos com a abordagem junto ao Keras, usando uma aplicação real com uma rede neural convolucional, dão evidências da flexibilidade, eficiência da coleta de dados, análise e validação dos dados da rede.*

Abstract. *The duration of the life cycle in deep neural networks depends on the data configuration decisions that lead to success in obtaining models. Analyzing hyperparameters along the evolution of the network's execution allows adapting the data, thus reducing the life cycle time. However, there are challenges not only in collecting hyperparameters, but also in modeling the relationships between these data. This work presents a provenance data based approach to address these challenges, proposing a collection mechanism with flexibility in the choice and representation of data to be analyzed. Experiments of the approach with Keras, using a real application provide evidence of the flexibility, the efficiency of data collection, the analysis and the validation of network data.*

1. Introdução

Nos últimos anos, a comunidade científica tem presenciado um aumento na popularidade de aplicações em técnicas de aprendizado profundo (*i.e., Deep Learning ou AP*) [Gharibi et al. 2019, Miao et al. 2017], incluindo as PINNs, redes neurais profundas guiadas pelas leis da Física na solução de problemas em ciência computacional [Raissi et al. 2017].

*Este trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, CNPq, FAPERJ e INRIA.

AP é o termo usado para denotar o problema de treinar redes neurais profundas e permitir inferências, como a rede neural convolucional (CNN) [Badan and Sekanina 2019]. O treinamento das CNNs busca pela configuração de uma série de hiperparâmetros (p. ex., taxa de aprendizado, *batch size*, número de épocas, *momentum* e *dropout*) que levem à validação dos resultados. As CNNs são particularmente sensíveis à definição de hiperparâmetros. Essas definições dependem da evolução do comportamento dos dados, só conhecido durante ou ao término da execução do treinamento. Adaptações a essas decisões são realizadas em um processo de tentativa e erro até se atingir a validação dos resultados [Zaharia et al. 2018]. A etapa de treinamento torna o ciclo de vida da CNN computacionalmente custoso, cansativo e propenso a erros. Apesar de soluções de otimização automática de hiperparâmetros, como o *Cloud Auto ML*[1], e [Badan and Sekanina 2019], a análise de dados junto à configuração é necessária. O uso de dados de proveniência [Moreau and Groth 2013] vem sendo proposto para contribuir para a flexibilidade da representação e análise de dados quando especializados para AP [Breck et al. 2019, Caveness et al. 2020]. Uma vantagem no uso de dados de proveniência é a padronização na representação de metadados e a derivação dos dados ao longo do treinamento da CNN por meio do PROV-DM do W3C [Moreau and Groth 2013].

A avaliação das diversas configurações de hiperparâmetros exige que seja feita a relação entre vários tipos de dados e metadados, p. ex., métricas como acurácia, dados do ambiente computacional utilizado para o treinamento da CNN [Zaharia et al. 2018]. Soluções de coleta automática de dados para análise tendem a ser pouco flexíveis quanto à escolha do que será coletado e principalmente sobre como esses dados são analisados. Soluções com flexibilidade na coleta e análise de dados de proveniência como em [Pina et al. 2019] exigem uma etapa de modelagem que pode ser uma barreira para a adoção de ferramentas de proveniência.

Este artigo apresenta uma abordagem centrada em dados de proveniência, propondo um mecanismo de coleta automática, porém com flexibilidade na escolha de dados a serem coletados e analisados. A abordagem foi baseada na CNNProv [Pina et al. 2019] e implementada por meio do Keras-Prov, uma extensão do Keras[2], que é uma API que executa sobre a plataforma de AP TensorFlow[3]. O objetivo do Keras-Prov é reduzir as ações que o usuário deve realizar para a análise de dados, em especial hiperparâmetros, durante o treinamento de CNNs. Ao adotar o padrão W3C PROV-DM, o Keras-Prov provê uma representação de dados pública, extensível e adotada em diversos domínios de aplicação, o que facilita a interoperabilidade. O Keras-Prov é capaz de rastrear as transformações de dados e conjuntos de dados relacionados aos hiperparâmetros e métricas do treinamento da CNN de modo automático. O armazenamento dos dados de proveniência e adaptações dos usuários em configurações em SGBDs permite que os dados ao longo do treinamento sejam analisados globalmente. Por ser possível estender o esquema da base de dados de proveniência, dados do domínio da aplicação podem ser modelados e acrescentados à base de proveniência ao serem coletados ao longo do treinamento. Experimentos realizados com a DenseED, uma aplicação real de CNN densa guiada por leis da Física, mostram a adequação do Keras-Prov para a análise de hiperparâmetros de CNNs e o baixo impacto no desempenho da API no Keras. Além desta introdução, este artigo contém outras quatro seções. A Seção 2 discute os trabalhos relacionados, a Seção 3 apresenta a abordagem Keras-Prov, a Seção 4 os experimentos e, finalmente, a Seção 5 conclui este artigo.

[1] `https://cloud.google.com/automl`
[2] `https://keras.io/`
[3] `www.tensorflow.org/`

2. Trabalhos Relacionados

Há um crescente uso de dados de proveniência em aprendizado de máquina (AM) [Schelter et al. 2017, Tsay et al. 2018, Miao et al. 2017]. No entanto, não foram encontradas abordagens que ofereçam funcionalidades do Keras-Prov (*i.e.*, utilização de dados de proveniência especificamente na fase de treinamento de CNNs, possibilidade de realizar a análise *online*, flexibilidade quanto à escolha do que vai ser coletado e a extensão para coleta de outros dados, independência de portal, e com baixa sobrecarga). O ModelHub [Miao et al. 2017] é um sistema para gerência do ciclo de vida de redes neurais profundas cujo objetivo é armazenar pesos de modelos em diferentes versões, com foco no AP. O ModelHub possui um modelo proprietário para representação dos metadados do treinamento da rede neural, o que dificulta a interoperabilidade. A ferramenta Runway [Tsay et al. 2018] tem como objetivo a gerência de artefatos de AM e AP, como modelos, dados ou experimentos. A solução permite rastrear o modelo e os dados desde o uso no treinamento até a implementação, facilitando a reprodutibilidade. No entanto, além de ser uma solução proprietária, é restrita à linguagem de programação Python. [Schelter et al. 2017] propõem uma ferramenta automatizada para extrair os metadados do modelo e apresentá-los com uma visualização interativa para auxiliar na comparação de experimentos. Dessa forma, essa solução concentra-se no rastreamento de metadados e na proveniência dos dados de experimentação de AM. Porém, a abordagem proposta não utiliza padrões de representação, como o W3C PROV, afetando a interoperabilidade. Essas soluções preveem o uso de AM para aplicações de negócios. Keras-Prov herda os benefícios da DfAnalyzer [Silva et al. 2018] para acomodar dados científicos junto à proveniência e execução em ambientes de processamento de alto desempenho, facilitando seu uso em PINNs.

3. Abordagem centrada em dados: Keras-Prov

O objetivo principal do Keras-Prov é reduzir o esforço de adaptação do código para captura de dados de proveniência (uma vez que o Keras não captura tais dados de forma nativa) durante o treinamento da CNN. O Keras foi escolhido por ser uma API de rede neural de código aberto muito utilizada e com documentação disponível. A arquitetura do Keras-Prov é apresentada na Figura 1, e é composta de três camadas: (i) Treinamento, (ii) Dados e (iii) Análise. A camada de *Treinamento* é onde o Keras é executado e interage com bibliotecas de AP como o TensorFlow e o Theano. É importante ressaltar que o Keras teve seu código modificado para capturar os hiperparâmetros mais comuns e enviar seus valores para o *Extrator de Proveniência*. O Keras-Prov identifica automaticamente as transformações de dados no código do usuário, os hiperparâmetros e seus valores utilizados em cada treinamento, assim como as métricas coletadas. O *Extrator* recebe os dados assincronamente e os grava na camada de *Dados*. O banco de dados de proveniência segue o esquema da Figura 3, que estende o Prov-Df [Silva et al. 2017]. A Figura 3 representa em termos dos conceitos Agente (*Agent*), Atividade (*Activity*) e Entidade (*Entity*) do W3C PROV as transformações de dados do processo de treinamento das CNNs e os hiperparâmetros. Além dos dados de proveniência, o modelo treinado também é armazenado na camada de dados. O *Visualizador de proveniência* gera uma representação visual do grafo de proveniência, de forma a facilitar a análise por parte do usuário. Tais componentes são extensíveis, de forma que diferentes bibliotecas de treinamento de CNNs podem ser acopladas para capturar e armazenar hiperparâmetros e outros dados de treinamento.

Para utilizar o Keras-Prov é necessário que o usuário defina no código quais os hiperparâmetros devem ser capturados e armazenados, com base na lista disponibilizada, definindo com o valor *True* ou *False*. O trecho de código na Figura 2 define dados de configuração da DenseED.

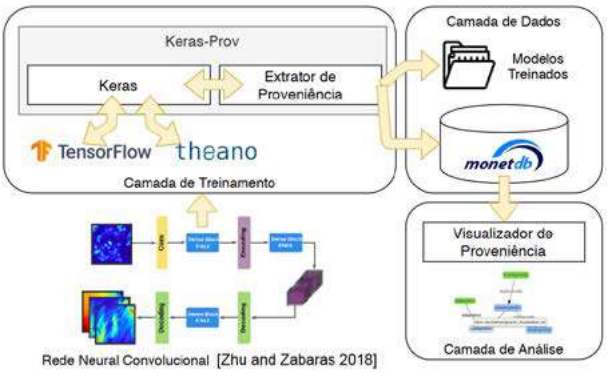

Figura 1: Arquitetura do Keras-Prov

```
tf1 = Transformation("Encoder")
tf1_output = Set("oEncoder", OUTPUT,
[ Attribute ("PADDING", TEXT),
 Attribute ("STRIDES", NUMERIC)])
 ...
tf3 = Transformation("DenseBlock")
tf3_input = Set("iDenseBlock", INPUT,
[ Attribute ("K", NUMERIC),
 Attribute ("L", NUMERIC)])
 ...
```

Figura 2: Código no Keras-Prov

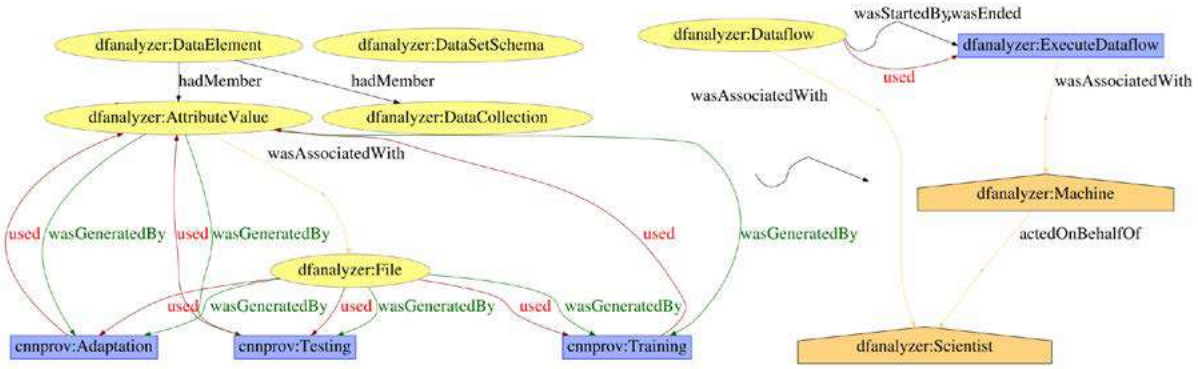

Figura 3: Modelo de Proveniência do Keras-Prov

4. Avaliação Experimental

Utilizamos a DenseED [Freitas et al. 2020] como estudo de caso do Keras-Prov. A DenseED é uma PINN que usa uma CNN densa, como um modelo substitutivo para viabilizar a quantificação de incertezas [Zhu and Zabaras 2018]. A DenseED tem como objetivo substituir o cálculo da migração reversa no tempo (MRT) por um modelo treinado. Desta forma o cálculo da MRT que teria que ser realizado para cada distribuição de probabilidade pode ser reduzido.

A instrumentação desta CNN utiliza as transformações *Treinamento*, *Adaptação* e *Teste*. *Treinamento* consome (*used*) o nome do otimizador, os hiperparâmetros taxa de aprendizado, número de épocas e número de camadas da rede, e produz (*wasGeneratedBy*) um conjunto de métricas que auxiliam na avaliação dos resultados obtidos durante o treinamento, p. ex., a acurácia, o valor da função de perda, o tempo decorrido e a data e hora do fim da execução de cada época. *Adaptação* consome (*used*) o conjunto produzido pela transformação anterior (*Treinamento*), um conjunto de dados com informações para a adaptação ocorrida e o conjunto de dados de saída contém a nova taxa de aprendizado, o valor da época e a data e hora em que a adaptação ocorreu, além de uma identificação para a adaptação. *Teste* provê dados sobre a avaliação do modelo de acordo com o conjunto de dados de treinamento e tem como saída os valores de acurácia e da função de perda. Porém, nesse caso, foram acrescentadas transformações para captura de informações acerca das camadas *encoder*, *decoder* e *dense block*.

O Keras-Prov foi avaliado sob duas perspectivas: (i) sobrecarga introduzida com a captura de proveniência e (ii) capacidade de consulta aos dados de proveniência. Com o código da DenseED adaptado para a utilização do Keras-Prov, foi realizado o treinamento variando-

se o número de épocas, taxa de aprendizado, *momentum*, *decay* e o otimizador escolhido. As execuções foram feitas com 20, 50 e 100 épocas, a taxa de aprendizado foi variada com os valores 0,0005, 0,001 e 0,002, o *decay* foi variado com os valores 0,0001 e 0,000001 e o *momentum* foi variado com os valores 0,5 e 0,9. Cada configuração de hiperparâmetros foi executada cinco vezes e a média do tempo de execução foi considerada. O tempo de execução foi medido a partir do início da aplicação até sua finalização, o Keras-Prov já estava inicializado antes da execução da aplicação e o MonetDB já estava com o *schema* definido (conforme Figura 3). O desvio padrão das cinco medições para as diferentes configurações de hiperparâmetros ficou entre 0,009 e 0,265. A medição da sobrecarga adicionada pelo Keras-Prov corresponde a um aumento de menos de 2,5% (no pior caso) sobre o tempo total de treinamento da CNN.

O potencial analítico do Keras-Prov resulta do processamento de consultas na base de proveniência. Foram executadas consultas para avaliar configurações de hiperparâmetros junto aos dados de métricas sobre o treinamento das CNNs para o usuário poder realizar os ajustes com mais segurança e confiança. Para as consultas da Figura 4 utilizamos a DenseED com o otimizador *Adam*, 20 épocas e taxa de aprendizado 0,001. A consulta (a) mostra "Qual o valor de perda para a DenseED com K=24 e L=4 no momento atual (época 200)?" e a consulta (b) mostra "Qual o valor da função de perda (*loss function*) de cada época para as quatro primeiras épocas da CNN DenseED?". À medida que o treinamento progride, o usuário continua realizando consultas à base. Por exemplo, ao se chegar em 200 épocas, o usuário decide finalizar a execução, pois esses valores não estão satisfazendo seus critérios. Nesse caso, o usuário altera o número de camadas no bloco denso para 8 e o treinamento segue com os novos dados que são armazenados na base de dados.

Qual o menor valor de perda e o número de camadas para valores de taxa de aprendizado = 0,001 e taxa de crescimento = 24?

época	valor de perda	num_camadas
200	0,012	64

Qual o valor de perda para cada época?

época	valor de perda
1	0,431
2	0,079
3	0,053
4	0,047

Figura 4: Consultas sobre o bloco denso = 4 junto às camadas *encoder*/*decoder*.

5. Conclusão

Este artigo tem como objetivo apoiar a fase de treinamento e otimização das CNNs ao registrar informações relevantes à análise de combinações de hiperparâmetros para reconfigurações. O Keras-Prov é uma extensão à API Keras, que é utilizada por muitas bibliotecas de AP (p. ex., TensorFlow e Theano). Por meio do Keras-Prov, o usuário não necessita de um grande esforço de adaptação de seu código para capturar dados de proveniência, ao mesmo tempo que possui flexibilidade para incluir novos dados a serem analisados e armazenados. O Keras-Prov foi avaliado com a CNN DenseED, um exemplo de uso de CNN por pesquisadores que não são da área de ciência da computação, uma situação que tende a aumentar e necessitar ainda mais de auxílio na análise e configuração de hiperparâmetros. Experimentos evidenciam a adequação do uso de proveniência nas atividades de análise ao longo do treinamento de CNNs, incluindo extensões para redes densas, contribuindo para um padrão de esquema que permite ao usuário consultar de forma integrada os dados de proveniência associados aos dados usados no treinamento. Como

trabalhos futuros, pretende-se estender o uso do Keras-Prov no apoio às redes neurais guiadas pelas leis da Física e quantificação de incertezas, que assim como em AP, requerem um processo analítico na criação de novos modelos.

Referências

Badan, F. and Sekanina, L. (2019). Optimizing convolutional neural networks for embedded systems by means of neuroevolution. In *TPNC 2019*, volume 11934, pages 109–121.

Breck, E., Polyzotis, N., Roy, S., Whang, S. E., and Zinkevich, M. (2019). Data validation for machine learning. In *Conference on Systems and Machine Learning (SysML).*

Caveness, E., GC, P. S., Peng, Z., Polyzotis, N., Roy, S., and Zinkevich, M. (2020). Tensorflow data validation: Data analysis and validation in continuous ml pipelines. In *Proceedings of the 2020 ACM SIGMOD*, pages 2793–2796.

Freitas, R. S., Barbosa, C. H., Guerra, G. M., Coutinho, A. L., and Rochinha, F. A. (2020). An encoder-decoder deep surrogate for reverse time migration in seismic imaging under uncertainty. *arXiv preprint arXiv:2006.09550*.

Gharibi, G., Walunj, V., Rella, S., and Lee, Y. (2019). Modelkb: towards automated management of the modeling lifecycle in deep learning. In *Int. Work. on Realizing Art. Intel. Synergies in Soft. Eng.*, pages 28–34. IEEE Press.

Miao, H., Li, A., Davis, L. S., and Deshpande, A. (2017). Towards unified data and lifecycle management for deep learning. In *2017 IEEE 33rd ICDE*, pages 571–582. IEEE.

Moreau, L. and Groth, P. (2013). Provenance: an introduction to prov. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 3(4):1–129.

Pina, D. B., Neves, L., Paes, A., de Oliveira, D., and Mattoso, M. (2019). Análise de hiperparâmetros em aplicações de aprendizado profundo por meio de dados de proveniência. In *XXXIV SBBD*, pages 223–228. SBC.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2017). Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*.

Schelter, S., Böse, J.-H., Kirschnick, J., Klein, T., and Seufert, S. (2017). Automatically tracking metadata and provenance of machine learning experiments. In *ML Systems workshop*.

Silva, V., de Oliveira, D., Valduriez, P., and Mattoso, M. (2018). Dfanalyzer: runtime dataflow analysis of scientific applications using provenance. *PVLDB*, 11(12):2082–2085.

Silva, V., Leite, J., Camata, J. J., De Oliveira, D., Coutinho, A. L., Valduriez, P., and Mattoso, M. (2017). Raw data queries during data-intensive parallel workflow execution. *FGCS*, 75:402–422.

Tsay, J., Mummert, T., Bobroff, N., Braz, A., Westerink, P., and Hirzel, M. (2018). Runway: machine learning model experiment management tool. In *SysML*.

Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S. A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M., Xie, F., and Zumar, C. (2018). Accelerating the machine learning lifecycle with mlflow. *IEEE Data Eng. Bull.*, 41:39–45.

Zhu, Y. and Zabaras, N. (2018). Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447.