Captura Automática de Dados de Proveniência de Experimentos de Aprendizado de Máquina com `Keras-Prov`

Débora Pina[1] **, Liliane Neves**[1]**, Daniel de Oliveira** [2]**, Marta Mattoso**[1]

[1]COPPE – Universidade Federal do Rio de Janeiro (UFRJ)

[2]Instituto de Computação - Universidade Federal Fluminense (IC/UFF)

{dbpina, lneves, marta}@cos.ufrj.br, danielcmo@ic.uff.br

Resumo. *Neste artigo apresentamos a Keras-Prov, uma extensão à biblioteca de aprendizado profundo Keras para prover dados de proveniência. A Keras-Prov captura, armazena e gerencia metadados e dados de proveniência de experimentos de aprendizado de máquina (ML), em especial de aprendizado profundo. A Keras-Prov identifica automaticamente as transformações de dados mais comuns, como, treinamento, teste, e adaptação, para capturar os dados de proveniência. A Keras-Prov flexibiliza a captura automática, permitindo que novos dados de proveniência sejam definidos, como valores adicionais de hiperparâmetros. À gerência de proveniência por meio do SGBD colunar MonetDB, Keras-Prov adiciona uma interface de monitoramento visual e um gerador de SQL para consultas analíticas aos dados durante a evolução do treinamento e a escolha de modelos. A análise de dados da Keras-Prov, durante o treinamento, subsidia decisões de sintonia fina de hiperparâmetros. A base de dados segue a recomendação W3C PROV, favorecendo a comparação, explicação e reprodução de tais experimentos de ML. A Keras-Prov é uma solução de código aberto e pode ser obtida em* `https://github.com/dbpina/keras-prov`.

1. Introdução

Os dados de proveniência [Freire et al. 2008, Moreau and Groth 2013] são uma solução natural para auxiliar usuários no registro de caminho de derivação de dados, metadados e parâmetros relevantes em experimentos que envolvem múltiplas transformações de dados. Os dados de proveniência são usados com sucesso para apoiar etapas de monitoramento, análise e reprodução em experimentos em múltiplos domínios, *e.g.*, bioinformática [Almeida et al. 2019], área de saúde [Fairweather et al. 2021], visualização [Fekete et al. 2020], *etc*. Seu uso durante o ciclo de vida de experimentos de Aprendizado de Máquina (do inglês *Machine Learning* - ML) vem ganhando importância [Souza et al. 2019]. O ciclo de vida de um experimento de ML pode ser considerado como um *workflow* baseado em dados, pois, a partir de dados brutos de entrada, produz um modelo de ML por meio de um fluxo de múltiplas transformações de dados e suas dependências.

Apesar de existirem inúmeros métodos de ML [Russell and Norvig 2020], o Aprendizado Profundo (*i.e.*, *Deep Learning* - DL) [Goodfellow et al. 2016] tem se mostrado um dos mais proeminentes. O desempenho de Redes Neurais Profundas (DNNs), treinadas a partir de um conjunto de dados de entrada, é bastante sensível à configuração de hiperparâmetros utilizada para o treinamento [Orr and Müller 2003]. É necessário fazer a sintonia fina de tais hiperparâmetros de forma complementar à sintonia automática. Para que o especialista possa ajustar os parâmetros durante o treinamento do modelo, é preciso ter acesso aos dados de "causa-efeito", *e.g.*, qual filtro foi aplicado ao modelo atual quando o valor de *dropout* se encontrava abaixo de um limite específico. Ao analisar os dados, o usuário pode decidir aceitar o modelo treinado ou selecionar uma nova configuração de hiperparâmetros, e, a partir desse

ponto, retreinar o modelo. O caminho de derivação representado por dados de proveniência favorece esse tipo de análise, quando os dados estão disponíveis para consulta durante o treinamento do modelo.

No contexto de experimentos de DL, os dados de proveniência, em conjunto com os metadados, têm grande potencial para apoiar a análise feita por humanos em relação às configurações de hiperparâmetros ou mesmo dados de treinamento. A avaliação dos vários hiperparâmetros requer que o usuário esteja ciente da relação entre muitos tipos de metadados, *e.g.*, os valores de hiperparâmetros escolhidos, dados de desempenho, configuração do ambiente, *etc*. Entretanto, para que o usuário seja capaz de realizar análises durante o treinamento, os dados de proveniência devem ser capturados e disponibilizados em tempo de execução, e adicionar dados de proveniência a experimentos de DL ainda é um problema em aberto em soluções de proveniência.

Ferramentas de visualização de fluxos de dados de DL, como o TensorBoard[1], utilizam *logs*. Comparar diferentes execuções demanda pré-processamento desses *logs*, assim como executar análises do caminho de derivação. Apesar de algumas ferramentas de DL disponibilizarem dados de proveniência, elas possuem representações proprietárias para os dados, o que torna difícil (e em muitos casos, impossível) interpretar e comparar experimentos de DL executados em diferentes ferramentas. Uma alternativa é utilizar uma solução agnóstica de domínio para capturar dados de proveniência nas ferramentas de DL [Pimentel et al. 2017]. Tais abordagens costumam ser acopladas a uma linguagem de programação específica (*e.g.*, Python), o que acaba limitando seu uso em muitas ferramentas de DL, ou são muito genéricas, exigindo assim um processo (muitas vezes) custoso de instrumentação do código.

Neste artigo, apresentamos a Keras-Prov, uma extensão à biblioteca Keras, para gerência e consulta a dados de proveniência em experimentos de DL. A Keras-Prov foi projetada para facilitar a análise de hiperparâmetros em experimentos de DL durante o treinamento. A Keras-Prov integra os dados de proveniência retrospectiva (também chamados de *r-prov*, *e.g.*, valores de hiperparâmetros, arquivos consumidos e produzidos, etc.) com dados específicos do domínio de DL. A Keras-Prov identifica, automaticamente no código do usuário, as transformações mais comuns como *Treinamento*, *Teste*, e *Adaptação*, oferecendo uma análise visual e com consultas analíticas aos dados. Os dados ficam disponíveis, durante a evolução do treinamento e a escolha de modelos, para auxiliar as decisões de sintonia fina. A Keras-Prov tem uma API para que os usuários consultem a base de dados de proveniência (que segue a recomendação W3C PROV) em tempo de execução, favorecendo a interpretação, análise e comparação desses experimentos de DL. O restante deste artigo está estruturado da seguinte forma. A Seção 2 mostra a arquitetura da Keras-Prov. A Seção 3 apresenta a demonstração, e a Seção 4 conclui o artigo.

2. Arquitetura da Keras-Prov

A arquitetura da Keras-Prov é composta por três camadas: (i) Núcleo do Keras-Prov, (ii) Camada de Dados e (iii) Camada de Análise, de modo a capturar, armazenar e analisar dados de proveniência em experimentos de DL, respectivamente. A biblioteca de treinamento de redes neurais profundas Keras[2] é estendida com essas três funções de gerência de proveniência. A Keras-Prov herda seu modo de processamento da biblioteca DfAnalyzer [Silva et al. 2018], *i.e.*, ela captura os dados de proveniência junto ao Keras, que são enviados de maneira assíncrona, sem interferir no desempenho do experimento de DL, para serem ar-

[1] `https://www.tensorflow.org/tensorboard`
[2] `https://keras.io/`

mazenados e consultados em processadores distintos da captura. Assim como na DfAnalyzer, a representação dos dados de proveniência na Keras-Prov segue a recomendação PROV do W3C [Moreau and Groth 2013] que é uma iniciativa para a representação de diferentes tipos de dados de proveniência de forma agnóstica ao domínio. Ao utilizar o W3C-PROV, a Keras-Prov gera uma cultura de análise de dados de proveniência, facilitando a comparação com resultados gerados, nesse padrão, em outras bibliotecas de ML. Além das vantagens de dados de proveniência para interpretar, reproduzir e adicionar qualidade ao dados do DL, o usuário alvo da Keras-Prov é a pessoa que especificou o experimento de DL, sendo o responsável pela sintonia-fina dos parâmetros. Assim, Keras-Prov se destaca por seu apoio analítico durante a geração do modelo.

Como o objetivo do W3C PROV é ser genérico, apresentamos uma especialização do modelo PROV, chamada DNNProv-Df em [Pina et al. 2021], para representar dados específicos do treinamento em experimentos de DL. Com base neste modelo, é possível: (i) rastrear épocas (*epochs*), taxa de aprendizado (*learning rate*), acurácia, função de perda (*loss function*), tempo de execução, *etc*. (ii) descobrir quais métodos de pré-processamento foram usados nos dados antes de treinar o modelo, (iii) monitorar o processo de treinamento e realizar um ajuste fino nos hiperparâmetros, (iv) descobrir quais arquivos foram gerados nas diferentes etapas de execução e (v) interpretar os resultados gerados. Por limitações de espaço, o modelo não é apresentado nesse artigo, mas pode ser obtido em [Pina et al. 2021]. A Figura 1 apresenta os principais componentes da arquitetura do Keras-Prov.

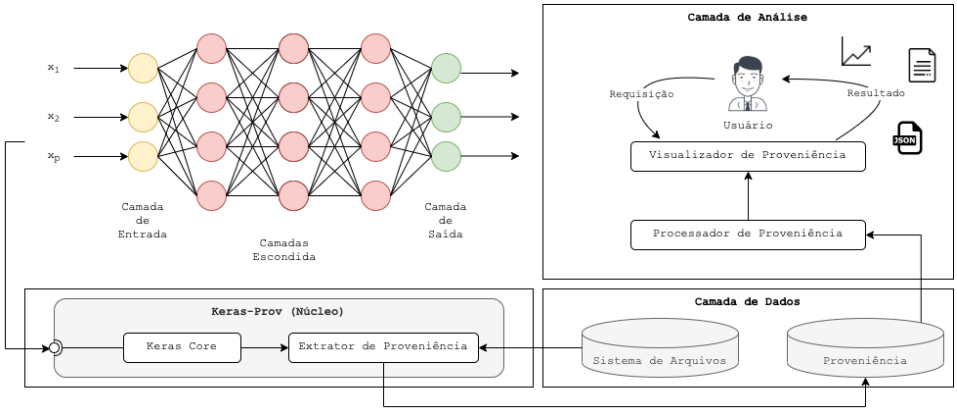

Figura 1. Arquitetura do Keras-Prov adaptada de [Pina et al. 2021]

O Núcleo da Keras-Prov é executado junto ao Keras padrão e interage com o *Extrator de Proveniência* (EP), que acessa e captura os valores dos hiperparâmetros em tempo de execução diretamente no Keras. O EP já se encontra acoplado à biblioteca Keras, dessa forma, o usuário não precisa instrumentar o seu código para capturar dados de proveniência. O usuário escolhe, entre as opções predefinidas, os dados de domínio (*datasets*) e hiperparâmetros a serem capturados. O EP extrai automaticamente os valores dos hiperparâmetros usados em cada treinamento. Uma vez capturados, os dados são gerenciados de forma assíncrona com o treinamento da rede neural. O EP acessa também o sistema de arquivos para obter os caminhos dos arquivos JSON que descrevem a rede neural e os dados de entrada da mesma. Esses caminhos dos arquivos, os valores de hiperparâmetros utilizados, métricas, etc., são enviados para o banco de dados de proveniência (na versão atual, instanciado no MonetDB[3]). Em seguida, o *Processador de Proveniência* consulta o banco de dados de proveniência e envia os resultados da consulta para o *Visualizador de Proveniência*, que gera uma representação visual do grafo

[3]`https://www.monetdb.org/`

de proveniência para análise de hiperparâmetros e é capaz de gerar gráficos (utilizando o Kibana) para análise de desempenho, *e.g.*, valores de *loss function* por época, *etc*. A Figura 2 apresenta a interface de consulta sobre os dados de proveniência capturados pela Keras-Prov. O grafo explicita os nomes das entidades (dados) representados como vértices e os nomes das transformações, representados por meio de arestas direcionadas que associam as entidades usadas como entrada da transformação para os dados de saída. Essa representação ajuda ao usuário definir filtros para que o SQL correspondente seja gerado automaticamente. A Figura 3 apresenta um exemplo dos gráficos de acompanhamento da evolução da acurácia do modelo para os dados de treinamento e validação ao longo das épocas (gerados pelo Kibana).

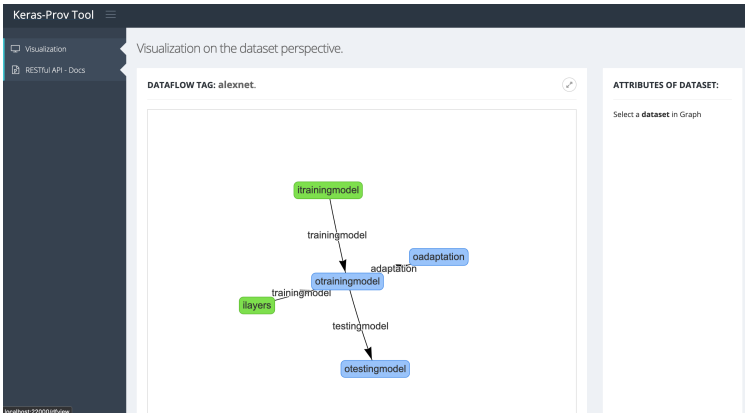

Figura 2. Interface da Keras-Prov

Figura 3. Gráficos extraídos do *dashboard* que apresentam a acurácia de treino e de validação durante a execução com otimizador Adam

3. Demonstração

A Keras-Prov tem código aberto e pode ser obtida no repositório `https://github.com/dbpina/keras-prov`. Além disso, se encontra disponível no mesmo endereço, um vídeo de demonstração da biblioteca e um guia inicial para usuários que desejam utilizar a Keras-Prov. A demonstração da Keras-Prov segue a especificação do experimento disponível na seção *Example* de seu repositório. A especificação tem como estudo de caso, a rede neural convolucional (CNN) AlexNet [Krizhevsky et al. 2012], que classifica imagens. A demonstração começa com o exemplo apresentado na Figura 4, onde o usuário define como *True* os hiperparâmetros de interesse. Em seguida, o usuário adiciona uma chamada ao método *provenance* em seu código. É importante notar que nenhuma instrumentação adicional é necessária. O método *provenance* faz a captura dos dados de proveniência. Esse método recebe um identificador para o experimento, um valor booleano que indica se há uma adaptação dos hiperparâmetros durante o treinamento (por exemplo, uma atualização da taxa de aprendizado) e a

```
hyps = {"OPTIMIZER_NAME": True,
   "LEARNING_RATE": True,
   "DECAY": True,
   "MOMENTUM": True,
   "NUM_EPOCHS": True,
   "BATCH_SIZE": True,
   "NUM_LAYERS": True}
```

```
model.provenance(
            dataflow_tag= "alexnet",
            adaptation=False,
            hyps = hyps)
```

Figura 4. Definição de hiperparâmetros de interesse com a Keras-Prov

lista de hiperparâmetros a serem capturados, conforme indicado pelo usuário no código-fonte do experimento de DL (Figura 4). Na demonstração, a execução da AlexNet é configurada com os seguintes valores de hiperparâmetros: (i) número de épocas = 30, (ii) taxa de aprendizado inicial = 0,001, (iii) tipo de ativação = *relu*, e (iv) otimizador = Adam. Além dos valores de hiperparâmetros, o conjunto de imagens OxfordFlowers[4] foi usado no treinamento. Durante o treinamento da DNN, os dados de proveniência são coletados e armazenados para consulta. O usuário pode então escolher entre (i) visualizar o grafo de proveniência (Figura 2), onde as transformações (Treinamento, Adaptação e Teste) e valores de hiperparâmetros podem ser visualizados, e (ii) visualizar gráficos de acompanhamento da evolução da acurácia do modelo para os dados de treinamento e validação ao longo das épocas (Figura 3). Os gráficos apresentados na Figura 3 são relativos à execução da AlexNet com otimizador Adam sem aplicação de filtros sobre as imagens. A partir deles, podemos observar uma discrepância no valor de acurácia para treino e validação. Essa análise pode indicar uma necessidade de sintonizar a taxa de aprendizado no próximo treinamento, ou até mesmo o número de épocas.

Durante o treinamento da rede neural, os usuários podem também submeter consultas diretamente ao banco de dados de proveniência. Essas consultas são definidas via interface gráfica (Figura 2) que gera o SQL correspondente e podem ser adaptadas de acordo com a necessidade do usuário. Por exemplo, na Tabela 1 apresentamos o resultado de uma consulta onde o usuário pode avaliar o impacto das adaptações da taxa de aprendizado na convergência do modelo. Embora a diminuição da taxa de aprendizado seja uma técnica conhecida, os valores registrados ajudam o usuário a identificar a causa das alterações ao analisar diferentes modelos. Outro exemplo é apresentado na Tabela 2, onde o usuário pode analisar se a aplicação do filtro que converte as imagens de entrada da AlexNet para a escala de cinza apresenta (ou não) melhora a acurácia. Nesse caso, a aplicação do filtro gerou uma acurácia de 0,37, enquanto que sem o filtro a acurácia foi de 0,59. Essa análise pode sugerir um novo ajuste. Para seguir a demonstração da Keras-Prov no evento, incentivamos os usuários a trazerem seus próprios experimentos de DL que já funcionem na biblioteca Keras.

Tabela 1. Análise parcial da Taxa de Aprendizado

Época	Tx. Aprendizado	Técnica
10	0.001	LRS
30	0.00025	LRS

Tabela 2. Impacto do filtro na acurácia

Filtro	Acurácia	Época
None	0.59	100
Gray-scale	0.37	100

4. Conclusão

A biblioteca Keras-Prov visa a apoiar a análise de configurações e adaptações de hiperparâmetros no treinamento de DNNs por meio da captura de dados de proveniência de interesse do usuário. A Keras-Prov permite capturar e armazenar dados de proveniência para

[4] https://www.robots.ox.ac.uk/~vgg/data/flowers/

consultas já em tempo de execução e gerar documentos que seguem a recomendação W3C PROV para ajudar na reprodução e interpretação do modelo da DNN. Além disso, ao adotar a W3C PROV, a Keras-Prov visa ser capaz de exportar dados de proveniência em um formato que permita a comparação com resultados obtidos em outras soluções de ML que também seguem a W3C. Como trabalhos futuros, começamos a estender a Keras-Prov para capturar dados característicos de Redes Neurais Informadas por Física (PINNs) [Raissi et al. 2019].

Agradecimentos

Trabalho realizado com apoio do CNPq, FAPERJ e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

Almeida, R. F., da Silva, W. M. C., Castro, K., de Araújo, A. P. F., Walter, M. E. T., Lifschitz, S., and Holanda, M. (2019). Managing data provenance for bioinformatics workflows using aprovbio. *Int. J. Comput. Biol. Drug Des.*, 12(2):153–170.

Fairweather, E., Wittner, R., Chapman, M., Holub, P., and Curcin, V. (2021). Non-repudiable provenance for clinical decision support systems. In *IPAW*, pages 165–182.

Fekete, J., Freire, J., and Rhyne, T. (2020). Exploring reproducibility in visualization. *IEEE Computer Graphics and Applications*, 40(5):108–119.

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1097–1105.

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

Orr, G. B. and Müller, K.-R. (2003). *Neural networks: tricks of the trade*. Springer.

Pimentel, J. F., Murta, L., Braganholo, V., and Freire, J. (2017). noworkflow: a tool for collecting, analyzing, and managing provenance from python scripts. *VLDB*, 10(12):1841–1844.

Pina, D., Kunstmann, L., de Oliveira, D., Valduriez, P., and Mattoso, M. (2021). Provenance supporting hyperparameter analysis in deep neural networks. In *IPAW*, pages 20–38.

Raissi, M., Perdikaris, P., and Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.

Russell, S. J. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson.

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

Souza, R., Azevedo, L., Lourenço, V., Soares, E., Thiago, R., Brandão, R., Civitarese, D., Brazil, E. V., Moreno, M., Valduriez, P., Mattoso, M., Cerqueira, R., and Netto, M. A. S. (2019). Provenance data in the machine learning lifecycle in computational science and engineering. In *WORKS*, pages 1–10. IEEE.