Avaliação da Alteração de Precisão e Leitura de Dados para Melhoria no Desempenho e Consumo de Energia de Algoritmos de Aprendizado de Máquina

Vitor Vieira, Felipe Bernardo, Bruno Schulze, Mariza Ferro

[1]Laboratório Nacional de Computação Científica – LNCC
Av. Getúlio Vargas, 333 – 25651-075 – Quitandinha, Petrópolis – RJ –Brasil

{vvieira,felipeb,schulze,mariza}@lncc.br

Abstract. *This work evaluated the influence of floating-point precision reduction and data reading techniques on Machine Learning algorithms in its training phase. The study is carried out using the Random Forest model. The aim is to analyze aspects of accuracy, execution time, and energy consumption to make better use of computational resources and reach solutions for a more ecological Machine Learning.*

Resumo. *Neste trabalho é avaliada a influência da redução da precisão do ponto flutuante e de técnicas de leitura de dados nos algoritmos de Aprendizado de Máquina durante a fase de treinamento. O estudo é feito com o uso do modelo de Floresta Randômica. O objetivo é analisar aspectos de precisão, do tempo de execução e do consumo de energia, afim de realizar um melhor uso dos recursos computacionais em busca de soluções para um aprendizado de máquina ecologicamente viável.*

1. Introdução

O Aprendizado de Máquina (AM) é uma importante subárea da Inteligência Artificial (IA). O AM teve enormes progressos na última década, com inúmeras aplicações práticas inseridas em atividades do dia a dia. A necessidade de modelos de IA e AM na solução de problemas complexos é evidente, porém, uma questão tem sido negligenciada, o consumo de energia desses algoritmos que muitas vezes utilizam alta capacidade computacional para o seu treinamento. Prevê-se que até 2040 entre 20% e 40% do consumo mundial de energia elétrica será atribuído a instalações de computação (Villani et al. 2018), e que atualmente *data centers* já são responsáveis por 3% do consumo global de energia [1].

Embora as aplicações de IA, incluindo o AM, tenham capacidade de gerar enormes benefícios para a sociedade, ela também dá origem a implicações éticas, sociais e ambientais que devem ser devidamente geridas. A principal razão para isso é que alguns modelos de AM são custosos para serem treinados, tanto em termos computacionais como na demanda de energia elétrica (Thompson et al. 2020). Além disso, a maioria das pesquisas nesta área se concentram em melhorar a precisão preditiva dos algoritmos, independente do custo econômico e ambiental atrelados a isso (Schwartz et al. 2020).

[1]independent.co.uk/climate-change/news/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html

Mediante a isso, é necessário uma mudança de paradigma na forma de executar algoritmos de AM em busca de um AM verde e sustentável. Para alcançar esse objetivo, este trabalho faz o estudo de caso para a fase de treinamento do algoritmo de Floresta Randômica (FR) avaliando a redução de precisão baseada em técnicas de software. São avaliados diferentes pontos de precisão, afim de encontrar uma configuração que diminua o consumo energético e o tempo, sem alteração na precisão preditiva do algoritmo. Além disso, são avaliadas formas de otimizar a leitura de dados para os algoritmos de AM.

2. Fundamentação Teórica

O AM pode ser definido como a arte de programar um computador para que ele possa aprender com os dados (Gron 2017) e pode ser dividido em dois tipos principais: supervisionado e não supervisionado. No aprendizado supervisionado é dado um conjunto de exemplos rotulados na forma (x_i, y_i), e o que se quer é produzir um modelo capaz de predizer o rótulo de novos dados. Se os rótulos sobre o qual se deseja fazer previsões assumem valores discretos tem-se a tarefa de classificação; para valores contínuos a tarefa é denominada regressão. Esse processo de indução de um classificador a partir de uma amostra de dados é denominado treinamento, onde os hiperparâmetros passam por ajustes até atingir uma configuração ideal que resulte na melhor precisão preditiva do modelo (erro ou acurácia). Com o treinamento finalizado e validado, o modelo de AM está pronto para a inferência.

Entre os algoritmos de AM supervisionado, o segundo mais utilizado pela comunidade de ciência de dados, segundo o relatório do Kaggle (Kaggle 2020), é a Floresta Randômica (Breiman et al. 1984). A FR é um algoritmo de *ensemble* do tipo *Bagging* (Breiman 2001), o qual reduz a variância das previsões combinando os resultados de várias Árvores de Decisão (AD) modeladas em diferentes subamostras do mesmo conjunto de dados. Ou seja, a FR pode ser descrita como um modelo formado por um conjunto de AD $h(X, y)$, onde y são vetores aleatórios amostrados de forma independentes, distribuídos igualmente em todas as árvores da floresta. O resultado é uma classe X com o maior número de votos entre todas as árvores consideradas (Ibañez et al. 2016). Este algoritmo é utilizado nas avaliações experimentais deste trabalho focado na sua fase treinamento e usando dados estruturados. Para isso foi utilizada a biblioteca *Scikit-learn* - Sklearn (Pedregosa et al. 2011), pois está entre as mais utilizadas atualmente para a implementação de algoritmos de AM (Kaggle 2020).

2.1. Ponto Flutuante

A aritmética de ponto flutuante (FP) ou mais conhecida como *float*, é a forma mais usada de aproximar a aritmética de números reais para realizar cálculos numéricos em computadores modernos. As abordagens para lidar com arredondamento, *underflows*, *overflows* ou "operações proibidas" (como $\sqrt{-3}$) são diferentes de uma máquina para outra. Essa falta de padronização dificultou a escrita de softwares numéricos confiáveis e portáveis. Os esforços de cientistas para padronizar essa aritmética resultou num padrão IEEE, sendo o IEEE-754 (Zuras et al. 2008) a versão mais recente.

O *Python* por padrão, define números de ponto de precisão flutuante dupla (*float64*), 11 bits para o expoente e 52 bits para mantissa, baseado no *double* do C[2],

[2]https://docs.python.org/3/library/stdtypes

que segue os padrão IEEE-754 (Zuras et al. 2008). O pacote *Numpy* aumenta as opções e limites que o Python determina para os *floats*, como o ponto de meia precisão (*float16*), definido com 5 bits para o expoente e 10 bits para mantissa; o ponto de precisão simples (*float32*) com expoente de 8 bits e 23 bits para mantissa. Por existir diferenças em bits de precisão entre os FP, é importante avaliar como cada um se comporta em relação a energia, tempo e predição no uso do algoritmo de FR.

2.2. Ferramentas para Manipulação dos Dados

Para implementação do algoritmo de FR com a biblioteca *SkLearn* (Pedregosa et al. 2011) é usada a biblioteca *Pandas* [3], utilizando o tipo de dado fornecido por ele, o *DataFrame*, que é uma estrutura de dados bidimensional rotulada com colunas de tipos diferentes. O *Pandas* é uma ferramenta de código aberto baseada em *Numpy* (Harris et al. 2020). Escrito principalmente em C, o *Numpy* é uma importante biblioteca no *Python* para cálculos numéricos e manipulação de matrizes. O *Pandas* é amplamente utilizada para manipulação e leitura de dados no *Sklearn* devido a sua facilidade de uso e por ser uma biblioteca rápida e eficiente.

Uma outra forma de se fazer a leitura de dados é utilizando a biblioteca *Dask* [4]. Com ele é usado o *Dask DataFrame*, que é similar ao *Pandas DataFrame*, porém, ele pode utilizar múltiplos núcleos para o processamento, diferente do *Pandas* que opera com apenas em um núcleo. O uso do *Dask* é indicado para manipular grandes conjuntos de dados e acelerar cálculos longos, visto que a paralelização beneficia essas operações.

Essas duas formas de leitura de dados são avaliadas neste trabalho.

3. Trabalhos Relacionados

A utilização de dados com menores precisões tem sido avaliada para melhorar o desempenho de diversas aplicações em computação científica (Haidar et al. 2017). O AM vem explorando com sucesso o uso de operações de ponto flutuante com precisão reduzida (por exemplo, *float16*) para economia de energia com uma perda mínima de precisão preditiva. Alguns trabalhos vêm apontando que para a IA é possível utilizar dados com menor precisão na fase de inferência, a qual pode ser implementada usando formato de dados com precisão menor, mas principalmente aplicados aos modelos de redes neurais (Zervakis et al. 2021). O trabalho de (Hashemi et al. 2017) investiga o balanço entre redução do uso de memória e de energia e as perdas de precisão para Redes Neurais Profundas (RNP). O trabalho demonstra que, na maioria dos casos, o dimensionamento de precisão pode trazer benefícios significativos ao custo computacional com diminuições muito modestas na precisão da rede. Também voltado para RNP, em (Jain et al. 2018) é avaliado o uso da redução de precisão como forma de reduzir os custos de processamento e armazenamento das redes para classificação de imagens. Além disso, é proposta uma abordagem de compensação do erro que é introduzido pela redução da precisão. Os resultados mostram boa redução no consumo de energia com perda de menos de 5% na precisão do classificador.

O trabalho de (Zervakis et al. 2021) faz uma ampla revisão da literatura sobre a aplicação de computação aproximada para área de AM, também apontando que a maioria

[3] `https://pandas.pydata.org/pandas-docs/stable/index.html`
[4] `https://dask.org`

dos trabalhos são voltados para modelos de redes neurais e durante a fase de inferência, pois é desafiador a aplicação deste tipo de abordagem na fase de treinamento.

O trabalho de (Holt and Sievert 2021) é focado no uso da biblioteca *Dask* para acelerar o treinamento do algoritmo Gradiente Descendente Estocástico (GDE). Foi usado um GDE específico que faz uso de GPUs para classificar imagens CIFAR10 [5]. Os resultados mostram que sem o uso do *Dask* o treinamento do algoritmo GDE pode levar mais de 2 horas dependendo da configuração escolhida, enquanto usando o *Dask*, pode acontecer uma redução no tempo de treinamento para até 45 minutos. Porém, neste trabalho não é avaliado o consumo de energia.

A maioria dos trabalhos encontrados na literatura se concentram em verificar a influência na redução da precisão para redes neurais com o objetivo de melhorar o desempenho e tempo, principalmente durante a fase inferência dos algoritmos. Ainda existem poucos trabalhos para algoritmos de AM com ponto flutuante de baixa precisão que avaliam o impacto dessa modificação no consumo de energia e durante a fase de treinamento, como avaliado neste trabalho.

4. Metodologia de Experimentos

A metodologia consiste em: i) executar o algoritmo de FR em diferentes configurações, variando o PF, tipos de bases de dados, e a forma de leitura desses dados; ii) aferir o tempo dessas execuções, resultados preditivos e consumo energético, afim de responder se é possível utilizar determinadas configurações, e quando. Todos os experimentos foram executados 10 vezes e calculada a média e o desvio padrão dos resultados.

Para a medição do consumo energético e do tempo é utilizada a ferramenta *Perf* e *Time*, ambas presentes no ambiente Linux. Para essas execuções foi utiliza a arquitetura baseada em um processador Intel core I7 8700 @3.2GHZ 6C e 12T, 64GB de RAM, Ubuntu 20.04 LTS, kernel 5.8.0-63-generic.

4.1. Configurações dos Experimentos

O algoritmo FR utilizado nos experimentos [6] foi implementado em linguagem *Python*, usando a biblioteca *Sklearn*. A configuração dos diferentes experimentos é descrita na Tabela 1. O algoritmo de FR é executado para as tarefas de classificação e de regressão, utilizando três conjuntos de dados sintéticos para o treinamento [7]. Os conjuntos têm 5 milhões de exemplos e 21 atributos. Os atributos foram definidos como totalmente categóricos, mistos (11 nominais e 10 numéricos), e totalmente numéricos.

Ainda, para os os experimentos com os conjuntos de dados mistos e numéricos é feita a variação do *float* (Tabela 1), entre *float16*, *float32* e *float64*. Neste caso, o uso do *Numpy* foi essencial, já que o *Python* define automaticamente qualquer dado para *float64*, e só foi possível alterar para as outras precisões devido ao seu uso. Essa variação não é possível de ser realizada com o conjunto de dados totalmente categórico, o qual só é avaliado na variação da forma de leitura de dados com *Dask* ou *Pandas*.

Assim, para cada tarefa ainda é utilizado o *Dask* para leitura dos conjuntos de dados, ou o *Pandas*. Na configuração onde será utilizada o *Dask* os dados terão que ser

[5]https://www.cs.toronto.edu/ kriz/cifar.html

[6]Disponível em: https://github.com/comcidis/WSCAD-WIC-VITOR-VIEIRA-2021

[7]Os conjuntos de dados foram criados com a ferramenta *Massive Online Analysis* (Bifet et al. 2010).

transformados novamente em *Pandas Dataframe*, visto que o *SkLearn* não aceita nativamente o tipo de dado *Dask Dataframe*. Para todos os experimentos os hiperparâmetros do algoritmo de FR foram definidos como max_depth = 10, que define a profundidade máxima de cada árvore da floresta, e n_jobs = -1 valor definido pela biblioteca quando se quer utilizar todos os núcleos do processador para construção da floresta. A definição desses valores dos hiperparâmetros foram baseados no trabalho de (Silva et al. 2021), escolhendo-se a configuração mais rápida.

Tabela 1. Configuração dos Experimentos

Float				Algoritmo
64 | 32 | 16				
Tarefa	Leitura dos Dados	max_depth	n_jobs	Floresta
Classificação e Regressão	Dask | Pandas	10	-1	Randômica

Vale ressaltar que os atributos categóricos são definidos como *object* pelo *Pandas*. Porém, de acordo com a sua documentação [8] é recomendável que se faça a mudança para um tipo de dado *category* para economizar memória. Já que o *Dask* DataFrame é baseado no *Pandas*, todos atributos categóricos dos conjuntos de dados executados neste trabalho foram definidos como *category*.

5. Resultados

O primeiro experimento verificou a influência da redução de precisão na tarefa de classificação. Os resultados são apresentados na Tabela 2 onde estão os valores medidos para energia (Joules), acurácia da classificação e o tempo de execução (segundos) para treinamento da FR. Os resultados utilizando diferentes pontos de precisão com a leitura de dados *Pandas* (colunas 16, 32 e 64 P) e *Dask* (colunas 16, 32 e 64 D) usando as bases de dados mista e numérica. É possível observar que utilizando a base de dados mista não ocorreu nenhuma mudança significativa no consumo de energia e no tempo de execução (menor que 0,1%) e nenhuma alteração na acurácia. Isso tanto para a acurácia usando o *Pandas* como o *Dask*. Já para a base somente com atributos numéricos (Tabela 2 - NUMÉRICA CLASSIFICAÇÃO), houve uma redução de 24% no consumo de energia quando reduzindo a precisão de *float64* para *float16*, com leitura de dados *Pandas* (64P e 16P ↓) e sem alteração na acurácia da classificação. Quando usando o *Dask*, a redução no consumo de energia foi de 21%, também sem redução da acurácia (64D e 16D ↓).

Ainda analisando os resultados para alteração da precisão, agora para a tarefa de regressão (Tabela 3). Os resultados com a base mista tiveram o mesmo comportamento da classificação, ou seja, sem alteração significativa nos valores para energia, MSE e tempo. Entretanto, para a base numérica é possível observar uma redução no consumo de energia de 26% quando reduzindo a precisão de *float64* para *float16*, com leitura de dados Pandas (64P - 16P ↓) e sem alteração no erro (MSE). Quando usando o Dask a redução no consumo de energia foi de 28,5% quando reduzindo a precisão de *float64* para *float16* (64D-16D ↓), também sem aumento no erro. Também houve redução expressiva no tempo execução, chegando a quase 200 segundos, como mostrado em destaque na Tabela 3.

Na Tabela 4 são apresentados os resultados comparativos para consumo de energia, precisão preditiva (acurácia ou MSE) da leitura com *Pandas* ou *Dask* para o conjunto

[8] https://pandas.pydata.org/pandas-docs/stable/user_guide/categorical.html

Tabela 2. Resultados da energia, acurácia e tempo da FR classificação - com (±desvio padrão) - utilizando diferentes pontos de precisão (float 16 - 32 - 64), com leitura de dados Pandas (P) ou Dask (D), base de dados mista ou numérica.

MISTA CLASSIFICAÇÃO						
	16 P	16 D	32 P	32 D	64 P	64 D
Energia (J)	27772,765 (±31,130)	27869,720 (±28,876)	27759,140 (±31,215)	27885,200 (±40,1875)	27778,430 (±28,289)	27868,385 (±25,264)
Acurácia	0,713 (±0)	0,713 (±0)	0,713 (±0)	0,713 (±0)	0,713 (±0)	0,713 (±0)
Tempo (S)	466,975 (±0,386)	465,440 (±0,204)	466,790 (±0,273)	464,995 (±0,151)	466,905 (±0,194)	465,120 (±0,197)
NUMÉRICA CLASSIFICAÇÃO						
	16 P	16 D	32 P	32 D	64 P	64 D
Energia (J)	**12188,295** ↓ (±21,248)	**12367,440**↓ (±23,384)	15476,035 (±11,113)	15585,960 (±14,439)	**16050,090** (±28,3805)	**15644,500** (±55,106)
Acurácia	0,993 (±0)	0,993 (±0)	0,994 (±0)	0,994 (±0)	0,994 (±0)	0,994 (±0)
Tempo (S)	197,600 (±0,833)	196,425 (±0,7825)	247,730 (±0,692)	245,850 (±0,346)	257,635 (±0,828)	245,730 (±0,507)

Tabela 3. Resultados da energia, MSE e tempo da FR regressão - com (±desvio padrão) - utilizando diferentes pontos de precisão (float 16 - 32 - 64), com leitura de dados Pandas (P) ou Dask (D), base de dados mista ou numérica.

MISTA REGRESSÃO						
	16 P	16 D	32 P	32 D	64 P	64 D
Energia (J)	148310,930 (±219,382)	148348,950 (±212,0315)	148236,525 (±193,093)	148251,095 (±222,347)	147977,975 (±195,1395)	148572,910 (±418,023)
MSE	0,357 (±0)	0,357 (±0)	0,357 (±0)	0,357 (±0)	0,357 (±0)	0,357 (±0)
Tempo (S)	2714,365 (±0,707)	2713,620 (±0,293)	2712,825 (±0,345)	2713,285 (±0,446)	2704,090 (±0,359)	2713,580 (±0,997)
NUMÉRICA REGRESSÃO						
	16 P	16 D	32 P	32 D	64 P	64 D
Energia (J)	**33097,700**↓ (±41,830)	**33240,160**↓ (±42,477)	45857,305 (±24,519)	45859,395 (±29,373)	**45751,645** (±51,201)	**46447,795** (±120,412)
MSE	0,001 (±0)	0,001 (±0)	0,001 (±0)	0,001 (±0)	0,001 (±0)	0,001 (±0)
Tempo (S)	**541,995**↓ (±0,304)	**539,535**↓ (±0,395)	724,480 (±0,505)	719,880 (±0,525)	**722,295** (±0,896)	**729,020** (±1,617)

de dados com apenas atributos categóricos. Como mencionado, para este tipo de dados não é possível avaliar a redução da precisão no *float*. É possível observar que não há diferença significativa entre a utilização do *Dask* ou o *Pandas* em nenhum dos resultados. Ainda, comparando o *Dask* e o *Pandas* na Figura 1 é apresentada a soma do tempo de execução e o consumo energético de todos os algoritmos executados neste trabalho, separados por execução com o *Pandas* e com o *Dask*. É possível observar que com ambas as bibliotecas usadas para leitura dos dados, avaliados com o algoritmo de FR não houveram mudanças significativas, tanto no consumo energético como no tempo.

6. Considerações Finais e Trabalhos Futuros

Neste trabalho foi possível perceber que a utilização de PF de menores precisões foi muito positiva para algumas execuções dos algoritmos. Todos os PF utilizados obtiveram um desempenho preditivo semelhante, porém, alguns com reduções significativas no consumo de energia e no tempo (resultados em destaque nas Tabelas 2 e 3). O destaque foi para redução de float64 para float16. Com relação ao uso do *Dask* para acelerar a leitura dos dados, o mesmo não se mostrou mais efetivo se comparado ao *Pandas*. Ambos

Tabela 4. Resultados comparativos Pandas e Dask para base categórica nas tarefas de classificação e regressão - com (± desvio padrão).

CATEGÓRICO CLASSIFICAÇÃO			
	Energia (J)	Acurácia	Tempo (S)
PANDAS	37154,210 (±88,831)	0,579 (±0)	620,695 (±3,334)
DASK	37212,365 (±106,361)	0,579 (±0)	623,760 (±0,311)
CATEGÓRICO REGRESSÃO			
	Energia (J)	MSE	Tempo (S)
PANDAS	266181,240 (±505,505)	0,528 (±0)	4531,000 (±1)
DASK	266607,755 (±552,096)	0,528 (±0)	4533,000 (±1,5)

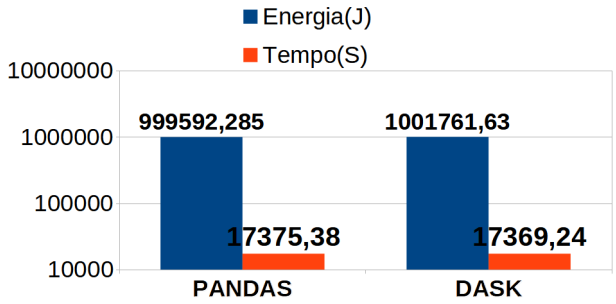

Figura 1. Soma da energia e tempo de todos os experimentos com *Pandas* e *Dask*, em escala logarítmica.

obtiveram o mesmo resultado no consumo energético e no tempo de execução. Esses resultados se mostraram promissores para o treinamento do algoritmo de FR, um dos mais utilizados no mundo. Porém, esses resultados podem não se repetir para algoritmos que utilizem outras formas de redução do erro, como a redução do gradiente no algoritmo baseado em AD XGBoost ou em algoritmos de Redes Neurais.

Em trabalhos futuros será avaliado o XGBoost e outros modelos que usem redução do gradiente para verificar a viabilidade da redução de precisão, ainda para a fase de treinamento, e também para a fase de inferência. Além disso, serão utilizados outros conjuntos de dados para investigar o impacto dos tipos de dados de treinamento para tarefas de classificação e regressão.

Agradecimentos

Os autores agradecem o apoio financeiro do LNCC/MCTI, CNPq e FAPERJ.

Referências

[Bifet et al. 2010] Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). Moa: Massive online analysis http://sourceforge. net/projects/moa-datastream. *Journal of Machine Learning Research (JMLR)*.

[Breiman 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

[Breiman et al. 1984] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

[Gron 2017] Gron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition.

[Haidar et al. 2017] Haidar, A., Wu, P., Tomov, S., and Dongarra, J. (2017). Investigating half precision arithmetic to accelerate dense linear system solvers. In *Proceedings of the 8th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*, pages 1–8.

[Harris et al. 2020] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

[Hashemi et al. 2017] Hashemi, S., Anthony, N., Tann, H., Bahar, R. I., and Reda, S. (2017). Understanding the impact of precision quantization on the accuracy and energy of neural networks. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 1474–1479.

[Holt and Sievert 2021] Holt, J. and Sievert, S. (2021). Training machine learning models faster with dask. *SciPy Conferences*.

[Ibañez et al. 2016] Ibañez, M. M., Ramos, F. M., and , A. R. C. (2016). Uso de redes neurais nebulosas e florestas aleatórias na classificação de imagens em um projeto de ciência cidadã. Master's thesis.

[Jain et al. 2018] Jain, S., Venkataramani, S., Srinivasan, V., Choi, J., Chuang, P., and Chang, L. (2018). Compensated-dnn: Energy efficient low-precision deep neural networks by compensating quantization errors. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6.

[Kaggle 2020] Kaggle (2020). State of data science and machine learning 2020. Technical report.

[Pedregosa et al. 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of ML Research*, 12:2825–2830.

[Schwartz et al. 2020] Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. (2020). Green ai. *Commun. ACM*, 63(12):54–63.

[Silva et al. 2021] Silva, G., Schulze, B., and Ferro, M. (2021). Performance and energy efficiency analysis of machine learning algorithms towards green ai: a case study of decision tree algorithms. Master's thesis, National Lab. for Scientific Computing.

[Thompson et al. 2020] Thompson, N. C., Greenewald, K., Lee, K., and Manso, G. F. (2020). The computational limits of deep learning.

[Villani et al. 2018] Villani, C., Bonnet, Y., schoenauer, m., berthet, c., levin, f., cornut, a. c., and Rondepierre, B. (2018). *For a meaningful artificial intelligence: towards a french and european strategy*. Conseil national du numérique.

[Zervakis et al. 2021] Zervakis, G., Saadat, H., Amrouch, H., Gerstlauer, A., Parameswaran, S., and Henkel, J. (2021). Approximate computing for ml: State-of-the-art, challenges and visions. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, ASPDAC '21, page 189–196, New York, NY, USA. ACM.

[Zuras et al. 2008] Zuras, D., Cowlishaw, M., Aiken, A., Applegate, M., Bailey, D., Bass, S., Bhandarkar, D., Bhat, M., Bindel, D., Boldo, S., et al. (2008). Ieee standard for floating-point arithmetic. *IEEE Std*, 754(2008):1–70.