DroidAutoML: uma Ferramenta de AutoML para o Domínio de Detecção de Malwares Android

Joner Assolin[2], Diego Kreutz[1], Guilherme Siqueira[1], Vanderson Rocha[2] Charles Miers[3], Rodrigo Mansilha[1], Eduardo Feitosa[2]

[1]Universidade Federal do Pampa (UNIPAMPA)
{DiegoKreutz,GuilhermeSiqueira.aluno,Mansilha}@unipampa.edu.br

[2]Universidade Federal do Amazonas (UFAM)
{joner.assolin,efeitosa}@icomp.ufam.br, vanderson@ufam.edu.br

[3]Universidade do Estado de Santa Catarina (UDESC)
charles.miers@udesc.br

Resumo. *Neste trabalho apresentamos a DroidAutoML, uma ferramenta de AutoML especializada para o domínio de detecção de malwares Android. A partir de um dataset de entrada e um pipeline de dados de quatro estágios, a DroidAutoML possibilita a geração automática de modelos preditivos otimizados. Os resultados iniciais indicam que a ferramenta é capaz de gerar modelos* muito bons*, chegando a 95% de recall.*

1. Introdução

Atualmente, existem diversas ferramentas relacionados ao tema de AutoML, como auto-sklearn[1], TPOT[2] e Google AutoML[3]. Tipicamente, essas ferramentas permitem obter artefatos testáveis e com um custo de tempo menor em comparação com abordagens manuais. Em particular, as ferramentas de AutoML reduzem significativamente a curva de aprendizagem, uma vez que eliminam a necessidade de diversos conhecimentos específicos da área de aprendizado de máquina. Entretanto, é importante ressaltar que essas ferramentas são tipicamente de propósito geral, i.e., não são desenvolvidas para gerar modelos otimizados para domínios específicos, o que impacta nas etapas de pré-processamento e engenharia de características.

Recentemente, sugiram soluções voltadas para domínios específicos, como reconhecimento facial [Yan et al., 2022] e veículos autônomos [Shi et al., 2021]. Em cada domínio podem existir desafios específicos, como compreensão do espaço do problema, operacionalização e avaliação da previsão [Karmaker ("Santu") et al., 2021] e personalização [Xin et al., 2021]. Desafios específicos podem ser melhor superados com o uso ferramentas de AutoML especializadas, que tiram proveito das particularidades do domínio de interesse para serem ajustados e gerarem modelos mais robustos e confiáveis que as ferramentas de AutoML de propósito geral. Adicionalmente, existem também desafios que podem ser considerados transversais, ainda não endereçados pelas ferramentas de AutoML existentes, como transparência e interpretabilidade [Xin et al., 2021] e recursos de depuração [Lee and Macke, 2020].

[1]https://automl.github.io/auto-sklearn/master/
[2]http://epistasislab.github.io/tpot/
[3]https://cloud.google.com/automl

As soluções de AutoML voltadas à um domínio específico possuem o diferencial de permitir o processamento especializado de dados, de modo a obter o máximo de informação relevante em relação ao domínio, principalmente através da criação de novas características com maior valor preditivo, por exemplo. É importante ressaltar que as soluções de AutoML, de propósito geral ou específicas, podem também possuir objetivos com escopo delimitado em diferentes dimensões. Por exemplo, recentemente, apresentamos uma solução de AutoML de propósito geral cujo objetivo principal é explorar a dimensão do tempo, ou seja, gerar modelos robustos e competitivos no menor intervalo de tempo possível. Os resultados revelaram que nossa solução permite gerar modelos preditivos de alta qualidade com baixo consumo de recursos computacionais [Siqueira et al., 2021], i.e., em pouco tempo (e.g., 8 vezes menos) quando comparada com outras soluções similares (e.g., TPOT).

Neste trabalho, apresentamos a DroidAutoML[4], uma ferramenta de domínio específico que abstrai a execução das etapas de limpeza de dados, engenharia de características, escolha de algoritmos e ajuste de hiper-parâmetros, identificando o algoritmo que melhor se ajusta ao problema de classificação de *malware* em Android. Adicionalmente, a DroidAutoML também implementa níveis de personalização, transparência, interpretabilidade e depuração que não são comuns nas ferramentas de AutoML de propósito geral, que funcionam essencialmente como "caixas pretas" (e.g., apenas apresentação das métricas finais aos usuários). Em sua versão atual, a DroidAutoML incorpora diferentes algoritmos em cada uma das etapas. Por exemplo, na engenharia de características a ferramenta incorpora três algoritmos (i.e., SigPID [Sun et al., 2016], *Robust Feature Generation* (RFG) [Moutaz, 2020], JOWNDroid [Cai et al., 2021]) sofisticados e otimizados de seleção de características para o domínio de detecção de *malwares* Android. Na etapa de configuração e otimização dos modelos utilizamos os algoritmos *K-Nearest Neighbors* (KNN), *Random Forest* (RF) e AdaBoost (ADB).

O restante deste trabalho está organizado como segue. Nas Seções 2 e 3 apresentamos a arquitetura e a implementação da DroidAutoML. A avaliação está na Seção 4, seguido de informações da demonstração e considerações (Seção 5).

2. DroidAutoML: Arquitetura e Implementação

A Figura 1 ilustra o *pipeline* de AutoML da DroidAutoML, com suas macro tarefas necessárias para a execução do fluxo de aprendizado de máquina: pré-processamento dos dados, engenharia de características, seleção de modelo e ajuste de modelo.

Na etapa de **pré-processamento dos dados** ocorre a correção de inconsistências e normalização de dados. Um *dataset* típico de detecção de *malwares* Android contém dados de múltiplas fontes (e.g., outros *datasets*, mercados de aplicativos) com diferentes níveis de verificação e padronização. Exemplos comuns de inconsistências incluem dados ausentes, tipos incorretos, redundantes e preenchidos incorretamente. Atualmente, a DroidAutoML realiza a detecção e correção de erros, como valores ausentes ou fora de limiares aceitáveis (e.g., não binários ou negativo).

Na **engenharia de características**, a tarefa principal é identificar aquelas características mais relevantes para o modelo de aprendizagem, com o objetivo de

[4]`https://github.com/Malware-Hunter/sf22-DroidAutoML`

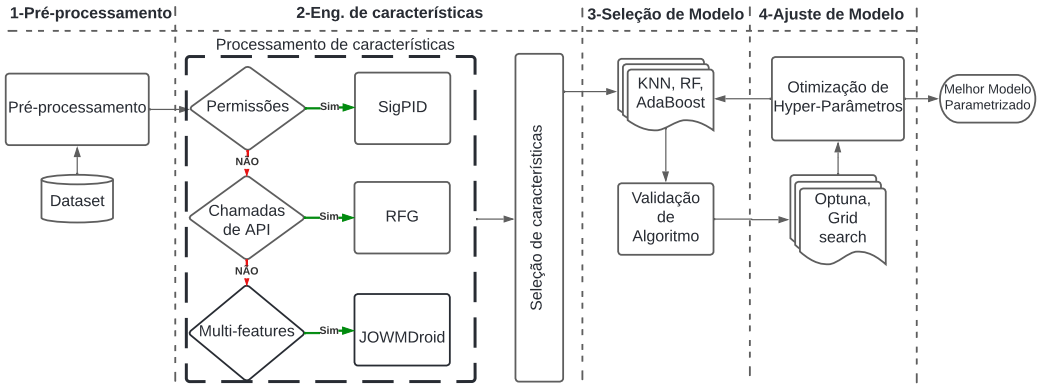

Figura 1. O *pipeline* da DroidAutoML.

aumentar a eficiência computacional do processo (e.g., reduzir o uso de memória). Por exemplo, na predição de *malwares* Android utilizando características como permissões, tipicamente, permissões como READ_SMS e SEND_SMS são relevantes, enquanto outras, como VIBRATE e SET_TIME, podem ser descartadas.

A redução de características é tipicamente realizada através de métodos como PCA (Análise de Componentes Principais) ou algoritmos de seleção de características como SelectKBest e RFE (*Recursive Feature Elimination*). Em domínios específicos, como detecção de *malwares* Android, temos também uma diversidade de métodos sofisticados e especializados, que observam diferentes aspectos particulares do domínio, como o impacto de diferentes tipos de características dos APKs de acordo com sua relevância [Sun et al., 2016, Cai et al., 2021, Moutaz, 2020]. Por exemplo, enquanto métodos mais generalistas (e.g., PCA) consideram características como a permissão INTERNET, devido ao seu grau elevado de recorrência, métodos específicos ao domínio, como SigPID [Sun et al., 2016], excluem essa característica por gerar ambiguidade ao modelo, pois a característica é tipicamente utilizada tanto por aplicativos malignos quanto benignos. Neste sentido, um dos principais objetivos da DroidAutoML é incorporar métodos de seleção de características específicos ao domínio, potencializando a construção de modelos menos ambíguos, eficazes e viáveis para *datasets* com grandes quantidades de amostras e características (e.g., 1 milhão de amostras e mais de 1 milhão de características). Na versão atual, ela incorpora três métodos específicos do domínio: SigPID, RFG e JOWNDroid.

A terceira etapa consiste na **seleção de modelo** de aprendizado de máquina considerando as alternativas conhecidas pelo sistema, o *dataset* e o problema de entrada. É importante observar que algoritmos de aprendizado de máquina são meta-heurísticas que oferecem soluções aproximadas para problemas considerados computacionalmente difíceis para se encontrar a solução exata. Portanto, a meta-heurística mais adequada pode variar dependendo do problema e dos dados disponíveis.

Atualmente, os algoritmos utilizados pela DroidAutoML são o KNN, RandomForest e AdaBoost. O RandomForest e KNN são algoritmos frequentemente utilizados e considerados entre os mais eficientes para detecção de *malwares* Android [Sharma and Rattan, 2021]. Enquanto que o KNN é considerado eficiente e eficaz para o reconhecimento de padrões, o RandomForest é um conjunto de árvores

de decisão para aprender e realizar previsões com *datasets* com elevada dimensionalidade. Finalmente, o AdaBoost é um dos algoritmos de *boosting* mais comuns e frequentemente utilizados para *ensemble learning* [Sagi and Rokach, 2018].

Finalmente, a última etapa é o **ajuste de modelo**, sendo realizado o processo de escolha e avaliação de valores para os parâmetros de entrada do algoritmo selecionado na etapa anterior. Para cada algoritmo é necessário fornecer uma combinação diferente de hiper-parâmetros (i.e., valores que controlam o processo de aprendizado de determinado modelo.

A DroidAutoML utiliza duas técnicas de otimização: a *optuna*, projetada para otimização automática de hiper-parâmetros, e a *grid search*, que cria e avalia modelos para cada combinação de hiper-parâmetros especificados, sendo guiada por alguma métrica de desempenho. Por exemplo, na detecção de *malwares* Android, *recall* é uma das métricas mais relevantes, pois indica a porcentagem de amostras maliciosas que foram classificadas corretamente.

3. Implementação

Para implementar a ferramenta utilizamos a linguagem de programação Python (versão 3.9.7) e bibliotecas de código aberto amplamente utilizadas em modelos de análise de dados e aprendizado de máquina, como Pandas (versão 1.3.4), para análise, limpeza e manipulação de dados, e Sklearn (versão 0.24.2), que fornece a implementação dos algoritmos KNN, RandomForest e AdaBoost. Utilizamos também bibliotecas especificas, como a Optuna (versão 2.10.0) para otimização automática de hiper-parâmetros, a Mlxtend (versão 0.19.0) para gerar regras de associação, a Matplotlib (versão 3.5.0) para criação de gráficos e visualizações de dados, e NumPy (versão 1.22.3) para operações matemáticas.

Na implementação da ferramenta, objetivando atingir um bom *design*, seguimos o princípio da responsabilidade única, isto é, uma classe deve ter apenas uma única responsabilidade e realizá-la de maneira eficaz. O objetivo é minimizar problemas de manutenção e facilitar o reuso de código [Martin et al., 2017]. Esse desacoplamento facilita também a evolução da ferramenta, oferecendo modularidade e simplicidade para a inclusão futura de novos métodos de seleção de características, algoritmos de aprendizado de máquina e técnicas de otimização de hiper-parâmetros.

Para reduzir problemas de usabilidade da ferramenta, com relação à interpretabilidade e depuração, cada etapa da execução é personalizada utilizando registros de eventos de dois tipos: `INFO` (informacional) e `CRITICAL` (problema de execução). As mensagens do tipo `INFO`, como representado por uma execução simples na Figura 2, permitem que o usuário acompanhe o que está acontecendo em cada etapa da execução da ferramenta, reduzindo o problema conhecido como "caixa preta" da maioria das ferramentas de AutoML. Já as mensagens `CRITICAL` permitem identificar problemas que precisam ser tratados, bastante útil para interpretabilidade, transparência e evolução contínua da ferramenta. Em termos de interpretabilidade e transparência, as mensagens `CRITICAL` podem identificar diferentes situações para os usuários, como erros de dados no estágio de pré-processamento ou falhas na execução de um método de seleção de características. Por exemplo, alguns métodos e mecanismos requerem certa especificidade em termos de dados de entrada (e.g.,

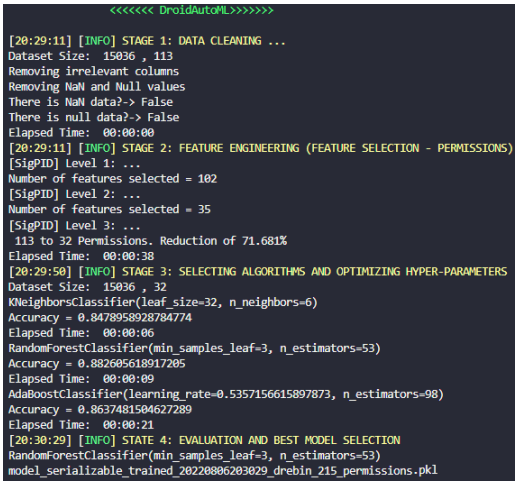

Figura 2. Exemplo de execução da ferramenta

não suportam certas características ou determinados tipos de dados). Informar o usuário sobre essas circunstâncias é fundamental em termos de transparência e interpretabilidade, reduzindo o risco de gerar modelos enviesados.

É importante destacar que a DroidAutoML apresenta detalhes da execução para cada etapa do fluxo do *pipeline* ilustrado na Figura 1. Por exemplo, a ferramenta informa aos usuários o tamanho do conjunto de dados, detalhes do processo de limpeza de dados, as particularidades (e.g., etapas e processos) de cada método de seleção de características (e.g., estágios dos três níveis do SigPID), relação de características selecionadas ao final da etapa de engenharia de características, relação de hiper-parâmetros e métricas dos modelos e a relação dos melhores modelos.

4. Resultados

Para avaliar a DroidAutoML, utilizamos o conjunto de *datasets* apresentados na Tabela 1. Ela mostra para *dataset* os tipos de características (permissões, chamadas de API e intenções), a quantidade de características (**#Car.**), a quantidade de amostras (**#Am.**), e a proporção de amostras benignas para malignas (**B:M**). Consideramos seis *datasets* distintos, sendo três (primeiros) derivados do AndroCrawl[5] e os outros três (últimos) do Drebin_215[6]. Por exemplo, o androcrawl_all representa o *dataset* original do AndroCrawl. Este contém todos os três tipos de características (permissões, chamadas de API e intenções) o que resulta em 82 características distintas. Desse conjunto de dados, nós derivamos outros dois *datasets* com as mesmas quantidade de amostras e proporção de amostras benignas para malignas, mas diferentes tipos e quantidades de características: o androcrawl_permissions, que contém contendo apenas 49 permissões; e o androcrawl_api_calls, que contém apenas 24 chamadas de API. É importante observar que os *datasets* derivados do AndroCrawl têm desbalanceamento entre amostras benignas e malignas (**B:M**) maior (cerca de 5 vezes) que o derivados do Drebin_215.

[5]https://github.com/phretor/ransom.mobi/blob/gh-pages/f/filter.7z

[6]https://figshare.com/articles/dataset/Android_malware_dataset_for_machine_learning_2/5854653

Tabela 1. *Datasets*.

Dataset	Tipos de Características	#Car.	#Am.	B:M
androcrawl_all	Permissões, Chamadas de API, Intenções	82	96.732	8,5:1
androcrawl_permissions	Permissões	49		
androcrawl_api_calls	Chamadas de API	24		
drebin_215_all	Permissões, Chamadas de API, Intenções	215	15.031	1,7:1
drebin_215_permissions	Permissões	113		
drebin_215_api_calls	Chamadas de API	73		

Usamos os seis *datasets* como entrada para DroidAutoML. DroidAutoML foi executada em um computador com processador Intel(R) Core(TM) i7-1185G7 3.00GHz 11th 8 Cores, 32GB RAM, HD, com sistema operacional *host* MS-Windows 10 64 bit. Para a execução da ferramenta foi utilizada uma máquina virtual VirtualBox[7] (versão 6.1.26 r145957 - Qt5.6.2) configurada com 4 vCPUs e 16GB RAM, executando GNU/Linux Ubuntu 20.04.3 LTS Desktop 64 bit. As seções subsequentes apresentam os resultados da redução de características (Seção 4.1) e modelos otimizados e selecionados (Seção 4.2) pela ferramenta para os *datasets* de entrada.

4.1. Seleção de Características

A principal atividade da etapa de engenharia de características da DroidAutoML é a seleção das características mais significativas de acordo com os *datasets* de entrada. Na Tabela 2 apresentamos um resumo dos métodos e respectivas reduções para cada conjunto de dados de entrada.

Tabela 2. Redução de dimensionalidade.

Dataset	Método	Redução (%)
drebin_215_permissions	SigPID	71,68
androcrawl_permissions	SigPID	87,76
drebin_215_api_calls	RFG	71,23
androcrawl_api_calls	RFG	12,50
drebin_215_all	JOWMDroid	82,30
androcrawl_all	JOWMDroid	88,89

Para os *datasets* androcrawl_permissions e drebin_215_permissions, ambos compostos apenas por permissões, a DroidAutoML selecionou automaticamente o método SigPID, que é especializado nesse tipo de características, e atingiu reduções de dimensionalidade de 71,68% e 87,76%, respectivamente. Para os *datasets* de chamadas de API, a DroidAutoML selecionou o método RFG, especializado nesse tipo de características, e atingiu reduções de 71,23 (drebin_215_api_calls)e 12,50 (androcrawl_api_calls). Finalmente, para os *datasets* com mais tipos de características, a DroidAutoML escolheu o método JOWNDroid e assim atingiu uma redução de dimensionalidade superior a 80% para ambos os conjuntos de dados.

É importante ressaltar também que a explicação para a redução de dimensionalidade menor para o *dataset* androcrawl_api_calls está relacionada ao número

[7] `https://www.virtualbox.org/`

(apenas 24) e relevância das chamadas de API nele contidas. Como pode ser observado na Tabela 1, o drebin_215_api_calls, por exemplo, contém 73 chamadas de API, sendo a maioria delas de pouco impacto para a detecção de *malwares* Android, o que se traduz na redução em mais de 70% da dimensionalidade do *dataset*.

4.2. Modelos e Métricas

Na Tabela 3 apresentamos um relatório de modelos e métricas para os seis *datasets* de entrada (Tabela 1). Como podemos observar, o modelo e hiper-parâmetros selecionados variam entre *datasets*. As métricas também diferem entre os conjuntos de dados. Por exemplo, o *recall* é de apenas 46% para o androcrawl_all, e chega a 95% para o drebin_215_all. Tempo está em total HH:MM:SS.

Tabela 3. Relatório sumarizado de modelos e métricas.

Dataset	**Modelo Selecionado**	**Acurácia**	**Precisão**	***Recall***	**F1_*Score***	**Tempo**
androcrawl_permissions	KNeighborsClassifier(leaf_size=55, n_neighbors=7)	71%	66%	66%	66%	00:03:55
drebin_215_permissions	RandomForestClassifier(min_samples_leaf=2, n_estimators=43)	86%	84%	64%	73%	00:01:42
drebin_215_api_calls	RandomForestClassifier(min_samples_leaf=2, n_estimators=43)	92%	88,9%	75%	82%	00:01:17
androcrawl_api_calls	RandomForestClassifier(min_samples_leaf=9, n_estimators=26)	85%	91%	52%	66%	00:02:05
androcrawl_all	KNeighborsClassifier(leaf_size=45, n_neighbors=6)	72%	54%	46%	50%	00:00:47
drebin_215_all	RandomForestClassifier(min_samples_leaf=2, n_estimators=44)	91%	93%	95%	94%	00:00:58

O Drebin_215 possui mais características e, também, um grupo maior de características mais significativas para a detecção de *malwares* Android. Por exemplo, no conjunto total de características, o JOWMDroid reduziu em 82,30% a dimensionalidade do Drebin_215 e em 88,89% do AndroCrawl.

5. Considerações Finais

Demonstração

A demonstração da ferramenta será realizada através de uma máquina virtual em ambiente hospedado em dispositivo próprio dos autores. As funcionalidades e saídas da ferramenta serão apresentadas através de: (i) apresentação dos parâmetros de configuração e execução; (ii) apresentação das etapas do *pipeline* de dados; (iii) detalhamento dos *datasets* de entrada; (iv) execução da ferramenta para cada um dos *datasets*; (v) detalhamento das mensagens da DroidAutoML durante a execução; e (vi) apresentação das saídas da ferramenta, i.e., *dataset* resultante da etapa de engenharia de características, relatório de métricas e modelo serializado.

Conclusão

Ao que sabemos, a DroidAutoML é a primeira ferramenta de AutoML especializada para o domínio de detecção de *malwares* Android. Atualmente, DroidAutoML incorpora três métodos de seleção de características e três algoritmos de aprendizado de máquina. A partir de um *dataset* de entrada e um *pipeline* de dados de quatro estágios, a DroidAutoML possibilita a geração automática de modelos preditivos otimizados. Os resultados iniciais indicam que a ferramenta é capaz de gerar modelos que podem ser considerados *muito bons*, chegando a 95% de *recall*, a principal métrica para detecção de *malwares* Android.

Como trabalhos futuros podemos elencar: (a) aumentar os níveis de personalização, transparência e interpretabilidade da ferramenta; (b) incorporar novos métodos de seleção de características; (c) incorporar mais algoritmos na etapa de seleção de modelos; (d) apresentar inferências sobre as métricas do modelo selecionado; (e) realizar uma avaliação extensiva de conjuntos de dados; e (f) comparar os resultados da ferramenta com modelos disponíveis na literatura.

Agradecimentos. Esta pesquisa foi parcialmente financiada, conforme previsto nos Arts. 21 e 22 do decreto no. 10.521/2020, nos termos da Lei Federal no. 8.387/1991, através do convênio no. 003/2021, firmado entre ICOMP/UFAM, Flextronics da Amazônia Ltda e Motorola Mobility Comércio de Produtos Eletrônicos Ltda. O trabalho foi realizado também com apoio da CAPES – Código de Financiamento 001.

Referências

Cai, L., Li, Y., and Xiong, Z. (2021). JOWMDroid: Android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters. *Computers & Security*, 100:102086.

Karmaker ("Santu"), S. K., Hassan, M. M., Smith, M. J., Xu, L., Zhai, C., and Veeramachaneni, K. (2021). Automl to date and beyond: Challenges and opportunities. *ACM Comp. Sur.*, 54(8).

Lee, D. J.-L. and Macke, S. (2020). A human-in-the-loop perspective on automl: Milestones and the road ahead. *IEEE Data Engineering Bulletin*.

Martin, R. C., Grenning, J., Brown, S., Henney, K., and Gorman, J. (2017). *Clean architecture: a craftsman's guide to software structure and design*. Number 31. Prentice Hall.

Moutaz, A. (2020). Automated malware detection in mobile app stores based on robust feature generation. *Electronics*, 9:435.

Sagi, O. and Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249.

Sharma, T. and Rattan, D. (2021). Malicious application detection in android—a systematic literature review. *Computer Science Review*, 40:100373.

Shi, X., Wong, Y. D., Chai, C., and Li, M. Z.-F. (2021). An automated machine learning (automl) method of risk prediction for decision-making of autonomous vehicles. *IEEE TITS*, 22(11):7145.

Siqueira, G., Rodrigues, G., Kreutz, D., and Feitosa, E. (2021). QuickAutoML: Uma ferramenta para treinamento automatizado de modelos de aprendizado de máquina. In *WRSeg21*.

Sun, L., Li, Z., Yan, Q., Srisa-an, W., and Pan, Y. (2016). SigPID: significant permission identification for android malware detection. In *11th MALWARE*, pages 1–8.

Xin, D., Wu, E. Y., Lee, D. J.-L., Salehi, N., and Parameswaran, A. (2021). Whither automl? understanding the role of automation in machine learning workflows. In *Proceedings of the CHI*.

Yan, C., Zhang, Y., Zhang, Q., Yang, Y., Jiang, X., Yang, Y., and Wang, B. (2022). Privacy-preserving online automl for domain-specific face detection. In *IEEE CVF*, pages 4134–4144.