フィーチャに基づく深層学習モデル設計方法の提案と評価

M2019SE006　太田　龍之介
指導教員　青山　幹雄

1　研究背景

近年，深層学習を活用したシステムの開発が広がっている．しかし，深層学習モデルの開発プロセスは，従来のソフトウェア開発プロセスとは異なっており，体系化されていない．そのため，従来の深層学習モデル開発では，要求を満たす学習モデルの生成には，しばしば開発者の試行錯誤が必要とされる[5][9]．このような発見的開発方法では，学習モデルを効率的，かつ，安定して開発することは困難である．

本稿では，データのフィーチャ(特徴量)に着目し，段階的に学習可能な学習モデル設計方法を提案する．これにより，学習をコントロール可能な安定した開発を実現する．

2　研究課題

本稿では，提案方法を適用する深層学習モデルとしてCNN (Convolutional Neural Network)を対象とする．研究背景を踏まえ次の 2 点を研究課題とする．

(1) フィーチャに基づく段階的に学習可能な深層学習モデル設計方法の確立．
(2) 提案方法の実データ適用による有効性と妥当性の評価．

3　関連研究

3.1　フィーチャ設計

機械学習モデルの精度を向上するためにフィーチャを設計する技術体系であり，human-in-the-loop の開発において反復的に行うことで，入力データを改善する[2][6]．しかし，従来のフィーチャ設計方法は属人的であり，機械学習システム開発においてソフトウェア工学の点で課題がある．

3.2　フィーチャ選択

フィーチャ設計方法の一つで，訓練データから学習に有効なフィーチャを選択することで，データセットの品質を向上させる[3]．フィーチャ選択方法の中でも，Wrapper Method では学習モデルの精度に基づき学習するフィーチャの組み合わせを評価する方法である．そのため，学習モデルの評価結果を次のフィーチャの選択にフィードバックしていると解釈できる．

3.3　フィーチャ抽出器を活用した深層学習モデル生成

深層学習では，事前に学習済みのモデルをフィーチャ抽出器として再利用することが可能であり，抽出器から抽出されるフィーチャに基づき学習を効率的に行う方法が提案されている．

学習の対象範囲をドメインと呼ぶ．本質的にはソフトウェアにおけるドメインと同じである．転移学習[7]では，あるドメインに対して学習済みのモデルを，フィーチャ抽出器として別のドメインに再利用することで，転移先のドメインに対して効率的な学習が期待できる．このようなフィーチャ抽出器を活用した学習方法では，抽出されるフィーチャに基づいた学習により学習の効率化を図っていると解釈できる．しかし，抽出されるフィーチャに基づき学習をコントロールすることで，学習モデル開発の効率化を図る方法は提案されていない．

3.4　インクリメンタル学習

従来の学習モデル生成方法では，新たな学習データに対して性能を維持して学習モデルを更新することは困難である．そこで，知識の蒸留(Distillation)を行いながらインクリメンタルに学習することで，反復ごとに新たなクラスのデータに対して学習済みモデルを適合させる方法が提案されている[1]．また，学習済みクラスと新たなクラスのデータのバイアスを軽減させることで，効率的にインクリメンタル学習を行う方法が提案されている[8]．しかし，これらの提案はモデルを新たなクラスに適応させる方法であるため，モデルの学習状態や収集したデータに基づいてモデルを改善することはできない．

4　アプローチ

4.1　反復型開発プロセスによる学習の制御

従来の発見的な開発方法では，開発者によって開発速度や生成可能な学習モデルの性能が異なるため，効率的，かつ，安定して開発することが困難である．本稿では，発見的開発を改善するために，学習を制御可能な段階的学習を行う反復型開発プロセスを提案する(図 1)．

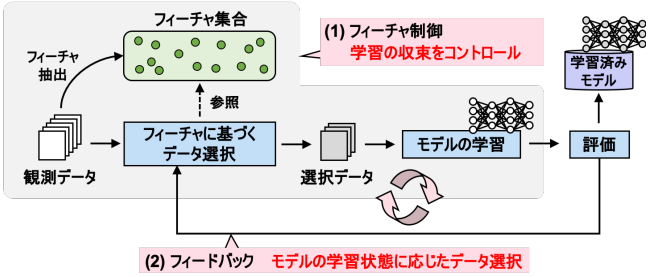

図 1 反復型開発プロセスのアプローチ

反復型開発プロセスでは，フィーチャ選択の Wrapper Method を応用する．事前に抽出する観測データのフィーチャと学習モデル評価結果に基づき選択された少量データを反復的に学習する．ここで，一般的に深層学習ではフィーチャが自動的に設計されるためにフィーチャ選択のプロセスは不要とされる．しかし，提案プロセスでは Wrapper Method を応用することで，フィーチャに基づく冗長な学習の回避を実現する．(詳細は 4.2 で述べる)．提案プロセスでは次の 2 点を実現する．

(1) フィーチャ制御

観測データから事前に抽出したフィーチャの集合に基づいて学習データを選択することで，学習時に獲得するフィーチャを制御する．これにより，学習の収束を制御する．

(2) フィードバック

反復ごとに学習の収束を評価し，その結果を次の学習データ選択へフィードバックする．これにより，学習の収束を改善する学習データ選択を可能とする．

フィーチャ制御とフィードバックを繰り返し実行することで，学習の収束を制御可能なインクリメンタル開発を実現する．結果として，要求を満たす学習モデルの効率的，かつ，安定した開発を可能とする．

4.2 フィーチャに基づく学習データ選択

本稿では，学習において類似度が高いフィーチャ集合に基づく学習では学習が偏り，冗長になると仮定する．特に，少量または偏ったデータの無作為な選択では，冗長になる可能性が高いと考える．そこで，学習時に獲得するフィーチャが均一になるように学習データの選択を制御して段階的に学習することで，冗長な学習の回避を図る．一方，学習の収束に有効なフィーチャはモデルの学習収束状態に依存すると考えられる．そのため，反復毎に学習の評価結果をフィードバックすることで，学習の収束に有効なフィーチャ選択を可能とする．

提案プロセスでは，冗長な学習の回避を図るために，フィーチャを局所的，かつ，段階的に選択することで，学習時に獲得するフィーチャを制御する(図 2)．

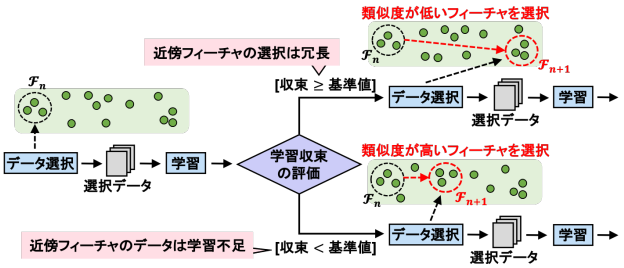

図 2 フィーチャに基づくデータ選択のアプローチ

提案プロセスの学習データ選択では，反復ごとに学習の収束を評価し，その評価結果に基づいて収束に有効なフィーチャを獲得可能な学習データを選択する．評価結果に基づいたフィーチャ選択のアプローチを以下に示す．ここで，n反復目に選択するフィーチャ集合を$\mathcal{F}_n$とする．

(1) n反復目の学習の収束が基準を満たす場合

$\mathcal{F}_n$を十分に獲得できたとみなす．これ以上，$\mathcal{F}_n$に対して近傍フィーチャを獲得することは冗長であると考えられるため，$\mathcal{F}_n$とは類似度が低いフィーチャを$\mathcal{F}_{n+1}$として選択する．

(2) n反復目の学習の収束が基準を満たさない場合

$\mathcal{F}_n$の獲得が不十分であるとみなす．再び$\mathcal{F}_n$に対して近傍フィーチャの獲得が必要であると考えられるため，$\mathcal{F}_n$とは類似度が高いフィーチャを$\mathcal{F}_{n+1}$として選択する．

このように選択したフィーチャ集合に基づくデータを学習データとして選択することで，偏ったフィーチャの獲得を防ぐ．これにより，反復ごとに学習の収束を改善する．

5 提案方法

5.1 フィーチャに基づく深層学習モデル開発プロセス

図 1 を詳細化した提案プロセスを図 3 に示す．

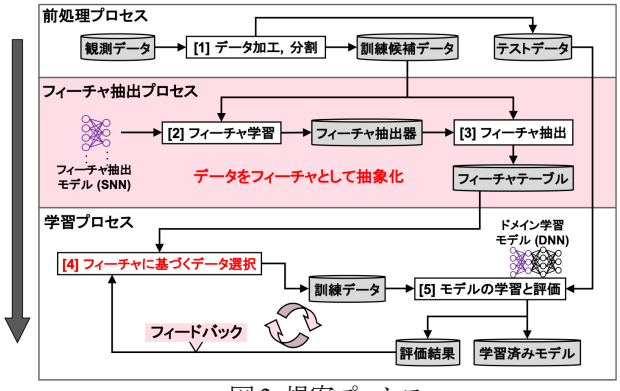

図 3 提案プロセス

提案プロセスでは，フィーチャ抽出プロセスによりデータを一度フィーチャとして抽象化することで，フィーチャに基づいたデータ選択を可能とする．これにより，学習時に獲得するフィーチャの制御が可能な学習を実現する．

(1) データ加工，分割

観測データから，訓練データとして選択する候補となる訓練候補データ及び評価に用いるテストデータを生成する．

(2) フィーチャ学習

フィーチャを設計する目的で，訓練候補データを学習する．生成されるモデルはフィーチャ抽出器として再利用する．

(3) フィーチャ抽出

各訓練候補データからフィーチャを抽出する(詳細は 5.2 で述べる)．データは抽出されたフィーチャと 1 対 1 で対応付けてフィーチャテーブルとして格納する．

(4) フィーチャに基づくデータ選択

フィーチャに基づき各反復で学習する訓練データを選択する(詳細は 5.3 で述べる)．反復ごとに学習時に生成されるフィーチャを制御することで，学習の収束の制御を図る．

(5) モデルの学習と評価

課題を解決する目的で，反復ごとに選択された訓練データを学習する．また，訓練誤差の収束速度と学習モデルの精度を測定する．収束速度の測定により，フィーチャがどの程度学習の収束に寄与したかを評価する(詳細は 5.4 で述べる)．評価結果は次のデータ選択にフィードバックされる．

5.2 フィーチャ抽出プロセス

図 3 のフィーチャ抽出プロセスを詳細化し，図 4 に示す．

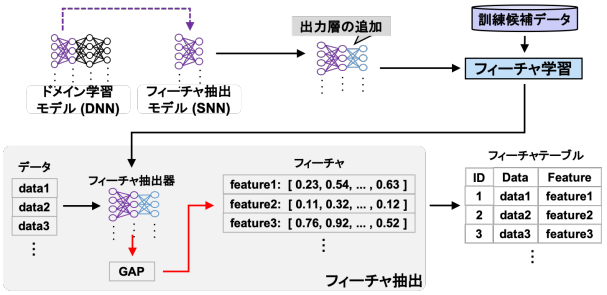

図 4 フィーチャ抽出プロセス

本稿では，フィーチャ抽出に用いる学習モデルをフィーチャ抽出モデル，また，学習対象範囲であるドメインに対し学習する学習モデルをドメイン学習モデルと呼ぶ．

フィーチャ抽出プロセスでは，ドメイン学習時に生成されるフィーチャを事前に設計することで，学習モデルの構成に則したフィーチャ制御を可能とする．ここで，フィーチャ抽出モデルには，ドメイン学習モデルの入力付近のニューラルネットワークを利用する．これにより，フィーチャ抽出時に生成されるフィーチャをドメイン学習時にも再現でき，後のフィーチャ選択を有効に機能させることを期待する．また，フィーチャはデータの特徴を厳密に表現できている必要はないため，フィーチャ抽出モデルには Shallow Neural Network(以下 SNN と略記)を利用し，フィーチャ学習のコストを削減する．抽出したフィーチャは，後の学習データ選択でフィーチャの類似度計算を行うために，Global Average Pooling(以下 GAP と略記)[4]によりベクトル化する．その後，データと抽出されたフィーチャを 1 対 1 で対応付けてフィーチャテーブルとして格納する．これにより，後に選択したフィーチャから学習データを参照可能にする．

5.3 学習データの選択方法

図 3 のフィーチャに基づくデータ選択を詳細化した学習データの選択方法を図 5 に示す.

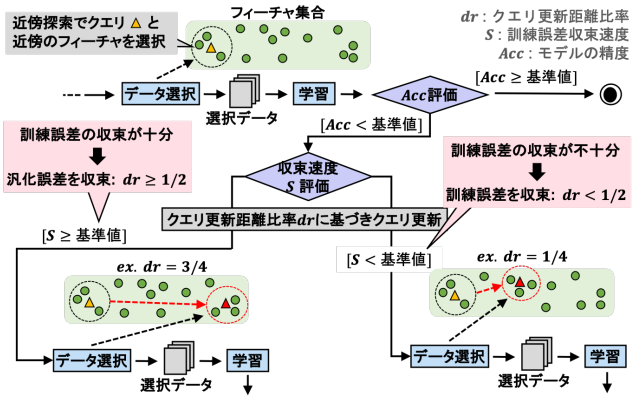

図 5 学習データの選択方法

提案方法の学習データ選択では, 図 2 のアプローチに則って各反復で学習の収束に有効なデータを選択することで, 学習の収束を制御する. モデルの評価結果から次の学習の収束に有効だと考えられるフィーチャと対応するデータを学習データとして選択する. フィーチャの選択は, あるフィーチャベクトルをクエリとする近傍探索で行う. 評価結果に基づいてクエリを更新することで, モデルの学習の収束に基づいたフィーチャ選択を可能とする. クエリの更新先は訓練誤差収束速度Sによって決定する. Sの詳細は 5.4 で述べる. ここで, クエリの更新先の設定を容易にするため, 式(1)で示すクエリ更新距離比率drを導入する. ここで, クエリから近傍i番目のフィーチャをf_iとする.

$$dr = \left(\frac{\text{クエリ更新先フィーチャの}\,i}{\text{探索対象のフィーチャの総数}}\right) \quad (1)$$

drが 0 の場合は最近傍, 1 の場合は最遠傍にクエリを更新することを意味する. Sに基づくクエリ更新方法を以下に示す.

(1) Sが基準を満たす場合

訓練誤差の収束が十分であるため, 訓練データの特徴を十分に獲得できたとみなす. そのため, 直前に獲得したフィーチャと近傍のフィーチャの獲得は冗長であるので, 次の学習では類似度が低いフィーチャを獲得する. したがって, drを 1/2 以上に設定し, クエリ更新を行う. 結果として, 獲得フィーチャが分散するため, 汎化誤差の収束が優先される.

(2) Sが基準を満たさない場合

(1)とは反対に, 訓練データが学習不足であるとみなし, 類似度が高いフィーチャを獲得する. したがって, drを 1/2 未満に設定することで, 再び訓練誤差の収束が優先される.

5.4 学習モデルの評価方法

図 3 のモデルの評価では, 図 5 の学習データの選択方法に則り, 学習モデルの精度と訓練データに対する学習の収束を評価する.

テストデータに対する正解率をモデルの精度Accとして評価する. Accが基準を満たした場合, モデルが要求を満たしているとみなし, プロセス終了とする.

一方, 本稿ではモデルが訓練データの特徴を十分に獲得できている場合, 訓練誤差の 0 への収束が速くなると仮定する. そのため, 訓練データのフィーチャを十分に捉えているかの判断指標として訓練誤差の収束速度を評価する. これにより, 学習の収束に基づいたフィーチャ選択を可能とする. しかし, 訓練誤差の収束速度を評価する方法は提案されていない. そこで, 本稿では訓練誤差収束速度Sを式(2)として定義する. ここで, 式(2)中の$epoch$は学習の最大エポック数, E_pはpエポックにおける訓練誤差を示す.

$$S = \sum_{k=1}^{\left\lfloor \frac{1}{n} epoch \right\rfloor} \frac{-a}{k}\left(E_{nk} - E_{nk-n+1}\right) \quad (2)$$

Sでは, 訓練誤差の収束速度を評価するために, 学習開始から最大エポックまでの訓練誤差の推移を評価する. 式(2)中のnはエポックの区間を示し, 各nエポック区間の訓練誤差推移の傾きの総和を収束速度として求める. また, 収束速度の評価として学習開始直後の推移が特に重要だと考えられるため, 各項にa/kの重みを付与し, 学習開始に近いエポック区間ほど収束速度が大きい値をとるように設定する. Sは訓練誤差の収束が速いほど高い値をとる.

6 プロトタイプの実装

6.1 実装環境

プロトタイプの実装環境を表 1 に示す.

表 1 ソフトウェアコンポーネント

コンポーネント	コンポーネント名	Version
OS	macOS Catalina	10.15.5
実装言語	Python	3.6.10
深層学習フレームワーク	PyTorch	1.4.0
データ加工ライブラリ	Pandas	0.22.0
可視化ライブラリ	Matplotlib	2.1.2
データベース	SQLite	3.28.0
近傍探索ライブラリ	Faiss	1.6.3

6.2 プロトタイプの構成

提案プロセスを評価するために実装するプロトタイプの構成を図 6 に示す.

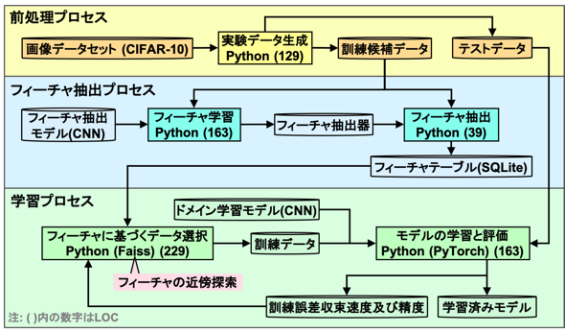

図 6 プロトタイプの構成

実装言語は Python, 深層学習フレームワークは PyTorch を採用した. フィーチャを格納するためのフィーチャテーブルには, Python から容易に操作でき, かつ, 処理性能の高い SQLite を利用した. また, 提案プロセスの学習データ選択ではフィーチャの近傍探索を行うため, 近傍探索ライブラリとして Faiss を利用した.

7 実データへの適用と評価

7.1 適用目的

実データに対してプロトタイプを適用し, 従来方法(無作為に学習データを選択し, 一括で学習する方法)と比較することで, 提案方法の有効性と妥当性を評価する.

7.2 適用対象

本稿では，提案方法を画像認識問題の 3 クラス分類に適用する．画像データセット CIFAR-10 のうち，automobile，bird，horse の 3 クラス画像を対象とする．各画像は，縦横 32px，チャネル数 3 のカラー画像とする．また，訓練データ数は 1 クラス 5,000 枚の計 15,000 枚，テストデータ数は 1 クラス 1,000 枚の計 3,000 枚を対象とする．

7.3 評価結果

提案方法と従来方法で学習データ数ごとに 3 回ずつ学習を行い，精度の平均値の推移を比較した結果を図 7 に示す．

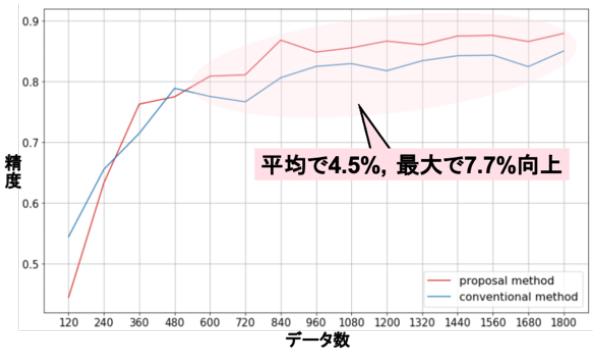

図 7 学習データ数ごとの精度の推移の評価

従来方法と比較して，提案方法の精度が特に学習データ数 600 以上で上回り，平均で 4.5%，最大で 7.7%向上した．

提案方法と従来方法の訓練誤差，汎化誤差の収束速度を式(2)で測定し，学習データ数毎の推移とばらつきを評価した結果を図 8 に示す．また，それぞれの収束速度の平均値と標準偏差を表 2 に示す．これらの結果から，従来方法と比較して，訓練誤差，汎化誤差ともに学習データ数毎に全体的に収束速度が向上した．図 8(c)と表 2 から，特に訓練誤差の収束速度が学習データ数の増加と共に標準偏差が顕著に減少した．

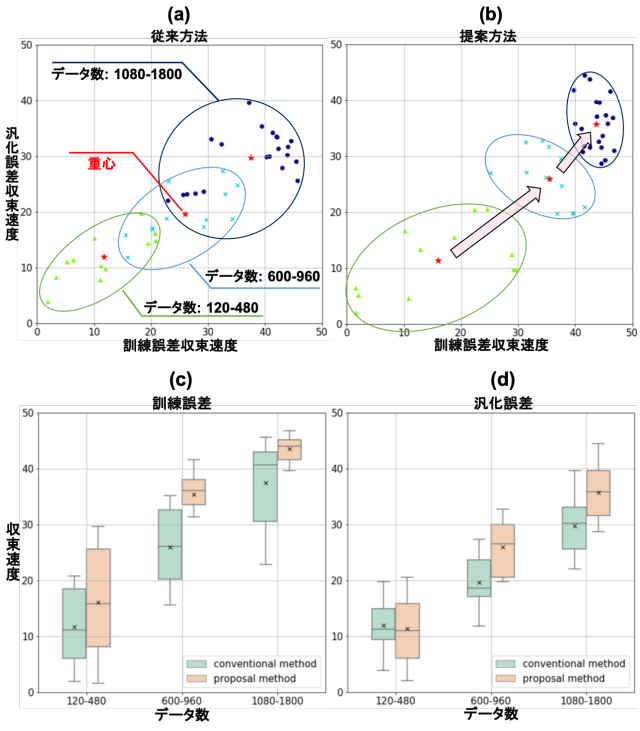

図 8 収束速度の推移とばらつきの評価

表 2 収束速度の平均値と標準偏差

比率 = (提案方法/ 従来方法)

	訓練誤差収束速度 (平均値, 標準偏差)			汎化誤差収束速度 (平均値, 標準偏差)		
データ数	120-480	600-960	1080-1800	120-480	600-960	1080-1800
従来方法	11.7,6.49	25.9,6.87	37.4,7.27	11.9,4.16	19.7,4.38	29.8,4.64
提案方法	16.0,10.5	35.4,4.33	43.6,2.07	11.4,5.90	26.0,4.82	35.7,4.63
比率	1.37,1.62	1.37,0.63	1.17,0.28	0.96,1.42	1.32,1.10	1.20,1.00

8 考察

8.1 評価結果の考察

図 7 から，提案方法では従来方法と比較して学習データ数が一定数以上の場合にモデルの精度を向上できた．提案方法の学習データ選択により学習時に獲得するフィーチャを制御することで，類似度が高いフィーチャ集合に基づく冗長な学習を回避できたと考えられる．また，図 8 の収束速度の推移とばらつきの結果は従来方法と比較して訓練誤差と汎化誤差ともに安定して減少できたことを示す．学習の収束結果のフィードバックをもとにフィーチャを制御したことで，学習の制御性が向上したといえる．

8.2 関連研究[1][8]との比較

関連研究[1][8]では，インクリメンタル学習において反復毎に学習モデルを新たなクラスのデータに対して最適化を行うが，学習の制御は行なっていない．提案方法では，反復毎にデータのフィーチャに基づき学習の制御を行うことで，モデルの学習の改善が可能である．

9 今後の課題

(1) 異なるフィーチャ選択方法による評価
(2) 他データセット，タスク，モデルへの適用と評価

10 まとめ

従来の発見的な深層学習モデル開発プロセスを改善するために，フィーチャに基づく段階的に学習可能な反復型開発プロセスを提案し，有効性と妥当性を示した．学習をコントロールしながらのインクリメンタルな開発を実現し，要求を満たすモデルの効率的，かつ，安定した開発が期待できる．

参考文献

[1] M. Castro, et al., End-to-End Incremental Learning, Proc. of ECCV 2018, LNCS Vol. 11205, Springer, Sep. 2018, pp. 233-248.
[2] G. Dong, et al., Feature Engineering for Machine Learning and Data Analysis, CRC Press, 2018, pp. 55-79.
[3] J. Li, et al., Feature Selection: A Data Perspective, ACM Computing Surveys, Vol. 50, No. 6, Article No. 94, Dec. 2017, 45 pages.
[4] M. Lin, et al., Network In Network, Mar. 2014, 10 pages, https://arxiv.org/abs/1312.4400.
[5] 丸山 宏, 城戸 隆, 機械学習工学へのいざない, 人工知能, Vol. 33, No. 2, Mar. 2018, pp. 124-131.
[6] S. Ozdemir, et al., Feature Engineering Made Easy, Packt, 2018.
[7] S. J. Pan, et al., Domain Adaptation via Transfer Component Analysis, IEEE Trans. Neural Networks, Vol. 22, No. 2, Feb. 2011, pp.199-210.
[8] Y. Wu, et al., Large Scale Incremental Learning, Proc. of CVPR 2019, June. 2019, pp. 374-182.
[9] D. Xin, et al., How Developers Iterate on Machine Learning Workflows, arXiv:1803.10311v2, May 2018.