インテル® End-to-End AI Optimization Kit による人工知能の高速化

インテルにより最適化されたソフトウェアで AI を大衆化

Jian Zhang インテル コーポレーション AI ソフトウェア・エンジニアリング・マネージャー

人工知能（AI）は、ヘルスケア、小売、製造業など、事実上すべての業界に革命をもたらしました。しかし、今日の AI ソリューションのほとんどは高コストであり、その利用は少数のデータ・サイエンティストに限定されています。これは複数の要因によるものです。第一に、最新のエンドツーエンドの AI パイプラインは複雑です。データ処理、特徴量エンジニアリング、モデルの開発、モデルのデプロイ、保守など、複数の段階が必要です。これらの段階の反復的な性質により、プロセスには時間がかかります。第二に、AI ソリューションの開発には、多くの場合、深い専門知識が必要です。これは、初心者や一般のデータ・サイエンティストにとって参入の障壁となります。第三に、人々は精度を高めるため、より大きく、より深いモデルを開発する傾向があります。これらの「過度にパラメーター化された」モデルは多大な計算量を必要とし、リソースに制約のある環境でのデプロイを妨げます。

インテル® End-to-End AI Optimization Kit（英語）は、エンドツーエンドの AI パイプラインをより高速に、シンプルに、アクセスしやすくし、どこでも、誰でも AI にアクセスできるようにするために開発されました。一般的なハードウェア上でハイパフォーマンスかつ軽量なモデルを効率良く提供する、エンドツーエンドの AI 最適化を実現する構成可能なツールキットです。このツールキットは、PyTorch* 向けインテル® エクステンション（IPEX）（英語）、TensorFlow* 向けインテル® エクステンション（ITEX）（英語）、インテル® AI アナリティクス・ツールキット（AI キット）など、インテルが最適化した一連のフレームワークに基づいて構築されています。また、ハイパーパラメーター最適化のため、SigOpt（英語）を統合します。インテル® End-to-End AI Optimization Kit は、データ準備、モデル最適化、およびモデル構築向けの独自のコンポーネントと機能も提供します。

エンドツーエンドの AI パイプラインのスケールアップとスケールアウトの効率を向上し、複雑なディープラーニング（DL）モデルの「夜間トレーニング」を可能にします。推論スループットが高く、リソース要件が低い軽量の DL モデルを提供します。また、エンドツーエンドの AI をよりシンプルにします。Click-to-Run ワークフローと SigOpt AutoML を使用してパイプラインを自動化し、データ処理と特徴量エンジニアリングの複雑な API を抽象化し、分散トレーニングを簡素化し、既存またはサードパーティーのマシンラーニング（ML）ソリューションやプラットフォームと簡単に統合できます。複雑な計算集約型の DL モデルを一般的なハードウェアで使用できるようにし、Smart Democratization Advisor（SDA）によって生成されたパラメーター化されたモデルを通じてビルトインの最適化されたモデルと、ニューラル・アーキテクチャー検索（NAS）テクノロジーで構築されたドメイン固有のコンパクト・ニューラル・ネットワークを提供します。これらすべてにより、データ・サイエンティストが AI にアクセスしやすくなります。

アーキテクチャー

さまざまなドメインの一般的なモデル (RecSys、CV、NLP、ASR、RL など) が、インテル® End-to-End AI Optimization Kit（**図 1**）の入力となります。これらのストックモデルは、重く、トレーニングに時間がかかり、チューニングや最適化が複雑です。モデルの種類に応じて、インテル® End-to-End AI Optimization Kit は、モデル・アドバイザーまたはニューラル・ネットワーク・コンストラクターのいずれかを使用してモデルを最適化します。最適化されたモデルは、FLOPS（1 秒あたりの浮動小数点演算数）とトレーニング時間がストックモデルの 1/10 で済み、精度の低下も同等か最小限に抑えられると期待されます。

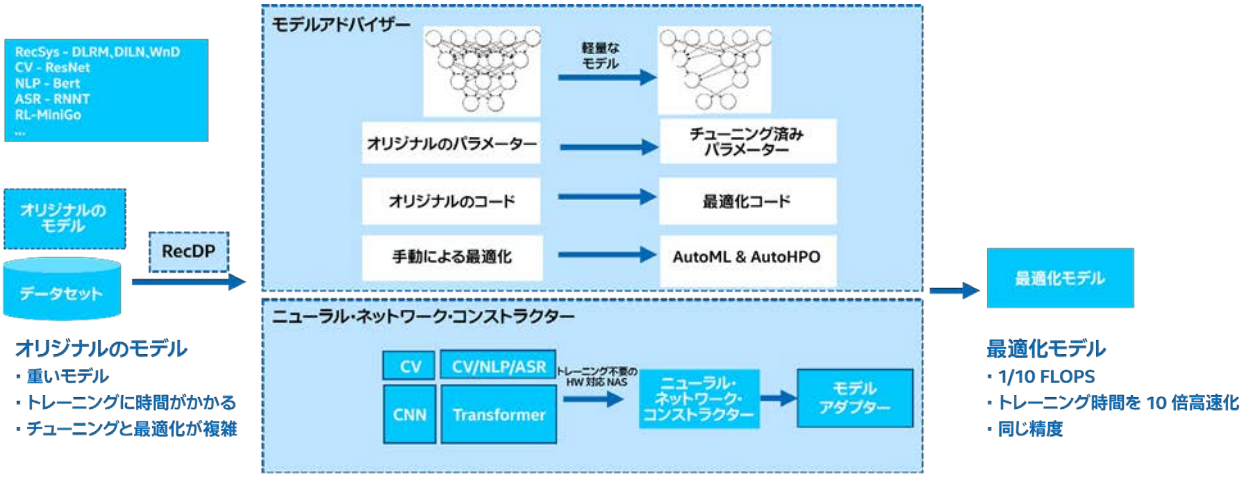

図 1. インテル® End-to-End AI Optimization Kit のアーキテクチャーとワークフロー

RecDP

RecDP は、PySpark* 上に構築された並列データ処理および特徴量エンジニアリング・ライブラリーであり、Modin（英語）などのデータ処理ツールに拡張可能です。主な特徴と機能は次のとおりです。

- 表形式のデータセット処理ツールキット
- Spark* プログラミングの複雑さを隠す抽象 API
- アダプティブ・クエリー・プランと戦略による最適化されたパフォーマンス
- ターゲット・エンコーディングやカウント・エンコーディングなどの一般的な特徴量エンジニアリング機能のサポート
- サードパーティー・ソリューションへの簡単な統合

RecDP は、パフォーマンスを向上するため、「遅延実行」を使用します。操作を融合し、データ統計のコレクションを活用して、データセットの不要なパススルーを回避します。これは、大規模なデータセットを処理する場合に重要です。RecDP は、Optimized Analytics Package for Spark* Platform（英語）によって提供される、ネイティブの単票形式の SQL エンジン機能を活用して、パフォーマンスを向上することもできます。

Smart Democratization Advisor（SDA）

SDA は、自動化を促進するユーザーガイド付きツールです。パラメーター化されたモデルによりビルトインのインテリジェンスを提供し、ハイパーパラメーター最適化（HPO）およびビルトインの最適化モデル（RecSys、CV、NLP、ASR、および RL など）に SigOpt を活用します。また、手動によるモデルのチューニングと最適化を変換して、AutoML と AutoHPO を支援します。

ニューラル・ネットワーク・コンストラクター

ニューラル・ネットワーク・コンストラクターは、ニューラル・アーキテクチャー検索テクノロジーに基づいています。定義済みのスーパーネットを使用して、特定のドメインのニューラル・ネットワーク構造を直接構築します。主な特徴と機能は次のとおりです。

- CV、NLP、ASR ドメインのモデルなど、マルチモデルのサポート
- 統合された Transformer ベースのスーパーネットを使用
- ハードウェア対応 NAS は、FLOPS やレイテンシーなどのメトリックをしきい値として使用してモデル・アーキテクチャーとモデルサイズを決定
- トレーニング不要の NAS は、候補を評価する際にトレーニング精度ではなく、ゼロコストのプロキシーメトリックを使用。トレーニング可能性、表現力、多様性、顕著性など、複数のネットワークの特徴が考慮されます
- モデルアダプターを活用して、ユーザーの運用環境にモデルをデプロイ。モデルアダプターは、微調整、知識蒸留、およびドメイン適応機能を提供する転移学習ベースのコンポーネントです

例

ここでは、ツールキットが DL 推奨モデル（DLRM）（英語）でどのように機能するか例を紹介します。これには、環境設定、データ処理、ビルトイン・モデル・アドバイザーとパッチコード、および最適化プロセス全体を開始する 1 行のコマンドが含まれます。

ステップ 1：環境設定

インテル® End-to-End AI Optimization Kit は AI キット上に構築されているため、パイプラインの実行に必要なソフトウェアはすでに利用可能な状態です。

```
# DockerFile
FROM docker.io/intel/oneapi-aikit
ENV http_proxy=http://proxy-prc.intel.com:912
ENV https_proxy=http://proxy-prc.intel.com:912

# SigOpt
RUN python -m pip install sigopt==7.5.0 --ignore-installed

# PyTorch* conda
RUN conda activate pytorch
RUN python -m pip install prefetch_generator tensorboardX onnx tqdm lark-parser

# PyTorch* 向けインテル(R) エクステンション
RUN python -m pip install intel_extension_for_pytorch==1.10.0 -f https://software.intel.com/
ipex-whl-stable psutil
RUN mkdir -p /home/vmagent/app
WORKDIR /home/vmagent/app
```

ステップ 2：RecDP による並列データ処理

次のステップでは、RecDP を使用してデータ処理を簡素化します。この例では、`Categorify()` と `FillNA()` の 2 つがチェーン操作であり、Spark* の遅延実行を使用してデータの不要なパススルーを減らしています。

```
from pyspark.sql import *
from pyspark import *
from pyspark.sql.types import *
from pyrecdp.data_processor import *
from pyrecdp.encoder import *
from pyrecdp.utils import *
import numpy as np

path_prefix = "file://"
current_path = "/home/vmagent/app/dataset/demo/processed/"
csv_file = "/home/vmagent/app/dataset/demo/criteo_mini.txt"

# 1. Spark* を起動して、データ・プロセッサーを初期化
t0 = timer()
spark = SparkSession.builder.master('local[*]') \
                            .config('spark.driver.memory','100G') \
                            .appName("DLRM").getOrCreate()
schema = StructType([StructField(f'_i{i}', IntegerType()) for i in range(0, 14)] )
df = spark.read.option('sep', '\t').option("mode", "DROPMALFORMED") \
               .schema(schema).csv(path_prefix + csv_file)
proc = DataProcessor(spark, path_prefix, current_path=current_path, spark_mode='local')

# 2. RecDP でデータを処理
CAT_COLS = list(range(14, 40))
to_categorify_cols = ['_c%d' % i for i in CAT_COLS]
op_categorify = Categorify(to_categorify_cols)
op_fillna_for_categorified = FillNA(to_categorify_cols, 0)
proc.append_ops([op_categorify, op_fillna_for_categorified])
df = proc.transform(df, name='dlrm')
t1 = timer()
print(f"Total process time is {(t1 - t0)} secs")
```

ステップ 3 : Smart Democratization Advisor（SDA）と AutoML

SDA は、手動による最適化を変換して AutoML を支援します。ビルトインモデルの定義済みパラメーターを提供することで、AutoML の作業時間を大幅に短縮できます。

```
model_info = dict()

# モデルの設定
model_info[“score_metrics”] = [(“accuracy”, “maximize”), (“training_time”, “minimize”)]
model_info[“execute_cmd_base”] = “python launch.py”
model_info[“result_file_name”] = “best_auc.txt”

# SigOpt の設定
model_info[“experiment_name”] = “dlrm”
model_info[“sigopt_config”] = [
    {‘name’:’learning_rate’,’bounds’:{‘min’:5,’max’:50},’type’:’int’},
    {‘name’:’lamb_lr’,’bounds’:{‘min’:5,’max’:50},’type’:’int’},
    {‘name’:’warmup_steps’,’bounds’:{‘min’:2000,’max’:4500},’type’:’int’},
    {‘name’:’decay_start_steps’,’bounds’:{‘min’:4501,’max’:9000},’type’:’int’},
    {‘name’:’num_decay_steps’,’bounds’:{‘min’:5000,’max’:15000},’type’:’int’},
    {‘name’:’sparse_feature_size’,’grid’: [128,64,16],’type’:’int’},
    {‘name’:’mlp_top_size’,’bounds’:{‘min’:0,’max’:7},’type’:’int’},
    {‘name’:’mlp_bot_size’,’bounds’:{‘min’:0,’max’:3},’type’:’int’}]
model_info[“observation_budget”] = 1
```

いくつかの設定可能なパラメーターのほかに、コードレベルの最適化が必要なケースがあります。すべてのビルトインモードに対して、キットは最適化されたモデルとパッチコードを提供します。この例では、IPEX と BF16 に加え、複数 CPU ノードでのモデルの収束を改善するオプティマイザーを使用します。

```
# フレームワークの最適化：  IPEX & BF16 を使用

# 最適化を埋め込む
-       m_curr = (m[i] if self.max_emb_dim > 0 else m)
-       EE = nn.EmbeddingBag(n, m_curr, mode=”sum”, sparse=True)
-       W = np.random.uniform(
-           low=-np.sqrt(1 / n), high=np.sqrt(1 / n), size=(n, m_curr)
-       ).astype(np.float32)
        # 大衆化、2 次元の疎行列/密行列を使用
+       W = np.random.uniform(
+           low=-np.sqrt(1 / n), high=np.sqrt(1 / n), size=(n, m)
+       ).astype(np.float32)

# オプティマイザーの最適化
- optimizer = torch.optim.SGD(dlrm.parameters(), lr=args.learning_rate) …)
```

ステップ 4 : エンドツーエンドのモデル最適化

最後に、数行のコードでエンドツーエンドのモデル最適化を開始します。

```
import sys
from e2eAIOK import SDA
sda = SDA(model =”dlrm”, settings=setting)
sda.launch()
```

インテル® End-to-End AI Optimization Kit は、次のような一般的なモデル向けの多くの Click-to-Run を提供しています。

- RecSys : **DLRM**（英語）、**DIEN**（英語）、**WnD**（英語）
- 自動音声認識（ASR）: **RNNT**（英語）
- コンピューター・ビジョン（CV）: **RESNET**（英語）
- 自然言語処理（NLP）: **BERT**（英語）
- 強化学習（RL）: **MiniGO**（英語）

パフォーマンス

テストは、4 ノードのクラスターで実施されました（**表 1**）。各ノードには、2 基のインテル® Xeon® Gold 6240 プロセッサーと 384GB のメモリーが搭載されており、ノードは 25GB イーサネットを介して接続されました。また、1TB HDD 1 台をデータドライブとして使用しました。

設定	説明
テスト実施日	2021 年 12 月
プラットフォーム	S2600WFT
CPU	インテル® Xeon® Gold 6240
ノード数	4
ノードごとの CPU 数	2 ソケット、ソケットごとに 18 コア、コアごとに 2 スレッド
メモリー	DDR4 デュアルランク 384G、12 スロット /32GB/2666MHz
ストレージ	1x 400GB インテル® SSD（SSDSC2BA400G3）OS ドライブ 1TB HDD（データストレージ）
ネットワーク	1x インテル® イーサネット・ネットワーク・アダプター X722、 1x インテル® イーサネット・ネットワーク・アダプター XXV710
マイクロコード	0x500002C
BIOS バージョン	SE5C620.86B.0X.02.0094.102720191711
OS/ ハイパーバイザー /SW	Fedora* 29 5.3.11-100.fc29.x86_64

表 1. システム構成

3 つの一般的な RecSys モデルで、ストックモデルと最適化されたモデルのパフォーマンスを比較しました（**表 2**）。最適化されたモデルは、3 つのワークロードすべてで、ETL、トレーニング、および推論を大幅に高速化しました（**図 2**）。

ワークロード	DLRM	WnD	DIEN
フレームワーク	PyTorch*	TensorFlow*	TensorFlow*
ライブラリー	oneMKL、oneCCL	oneMKL、oneCCL	oneMKL、oneCCL
データセット（サイズ、形状）	Criteo（英語）	outbrain（英語）	categoryFiles（英語）
精度（FP32、INT8、BF16）	BF16	-	-
KMP_AFFINITY	granularity=fine,compact,1,0	granularity=fine,compact,1,0	granularity=fine,compact,1,0
numactl	ソケット・バインディング	ソケット・バインディング	ソケット・バインディング
OMP_NUM_THREADS	20	16	4

表 2. テスト設定

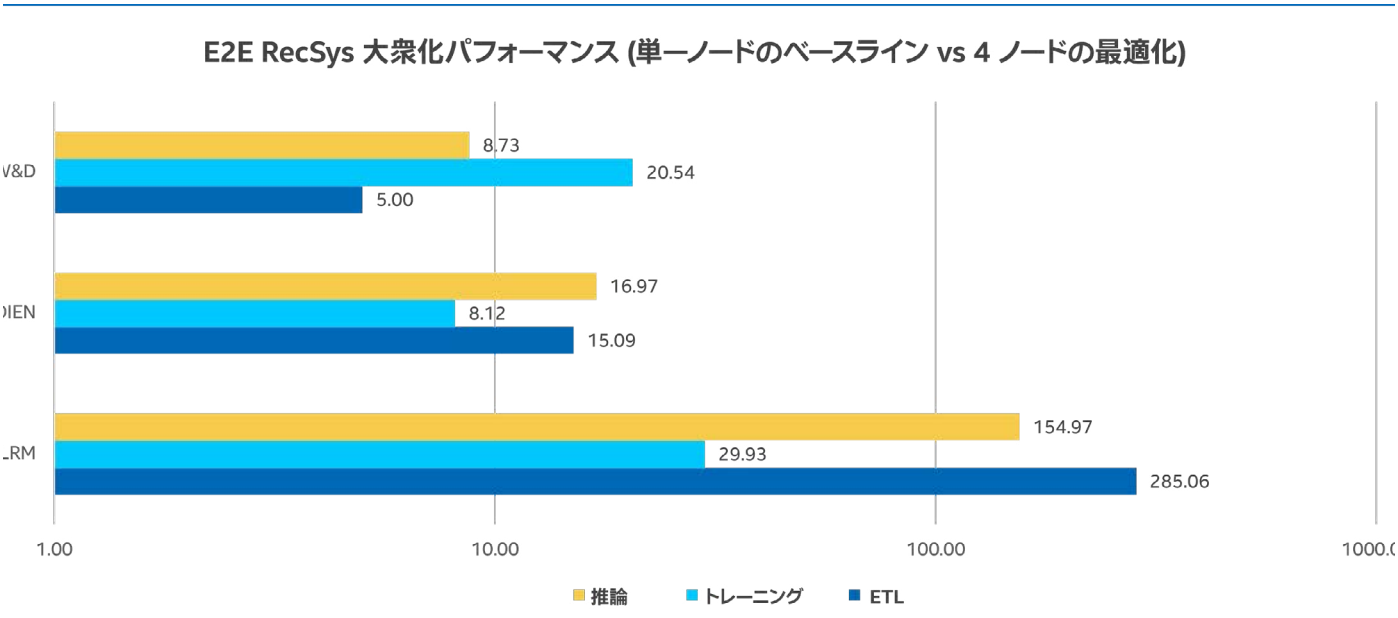

図 2. インテル® End-to-End AI Optimization Kit による RecSYS のスピードアップ。ベースライン構成：1 ノード、2x インテル® Xeon® Gold 6240 プロセッサー（18 コア）、インテル® ハイパースレッディング・テクノロジー有効、インテル® ターボ・ブースト・テクノロジー有効、384GB（12 スロット /32GB/2666MHz）メモリー、BIOS SE5C620.86B.0X.02.0094.102720191711（ucode:0x500002C）、Fedora* 29（5.3.11-100.fc29.x86_64）、PyTorch*、TensorFlow*、Spark*。最適化構成：4 ノード、PyTorch* 向けインテル® エクステンション、TensorFlow* 向けインテル® エクステンション、Horovod*、変更した DLRM、WnD、および DIEN ワークロードを除いて同じハードウェアとソフトウェア構成。

まとめ

インテル® End-to-End AI Optimization Kit は、一般的なハードウェア向けのハイパフォーマンスかつ軽量なモデルを提供する構成可能なツールキットであり、AI の大衆化に役立ちます。いくつかの重要な最適化を活用しています。

- RecDP による並列データ処理
- インテルにより最適化されたトレーニング・フレームワーク
- レイヤー数が少なく、分散メモリー型並列コンピューターで通信オーバーヘッドを軽減する軽量モデル
- 大きなバッチサイズで高速に収束するオプティマイザー・チューニング (DLRM)
- 特徴量エンジニアリング（埋め込みテーブルとエンコード）の最適化

このツールキットは、一般的なモデルで良好な結果をもたらします。プロジェクトで使用する場合は、https://github.com/intel/e2eAIOK/（英語）を参照してください。