インテル® ニューラル・コンプレッサーによる PyTorch* 推論の高速化

精度を損なわずに AI 推論をスピードアップ

Feng Tian インテル コーポレーション AI テクニカルリード
Haihao Shen インテル コーポレーション AI アーキテクト
Huma Abidi 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
Chandan Damannagari インテル コーポレーション AI ソフトウェア製品マーケティング・ディレクター

インテル® ニューラル・コンプレッサー（英語）は、CPU と GPU 上でディープラーニング（DL）の推論のモデルサイズを小さくして高速化する、オープンソースの Python* ライブラリーです（**図 1**）。量子化、プルーニング、知識蒸留などの一般的なネットワーク圧縮技術に対して、複数の DL フレームワークにわたる統一されたインターフェイスを提供します。このツールは、ユーザーが最適な量子化モデルを素早く見つけられるように、精度を意識した自動チューニングをサポートしています。また、事前に定義された目標を使用してプルーニングしたモデルを生成するため、異なるウェイト・プルーニング・アルゴリズムを実装し、教師モデルから生徒モデルへの知識の蒸留を支援します。インテル® ニューラル・コンプレッサーは、フレームワーク間の相互運用性を高めるため、ONNX* ランタイムに加えて、TensorFlow*、PyTorch*、MXNet* などのフレームワーク向けに API を提供しています。ここでは、PyTorch* モデルでこのツールを使用する利点に注目します。

© Intel Corporation. Intel、インテル、Intel ロゴ、その他のインテルの名称やロゴは、Intel Corporation またはその子会社の商標です。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

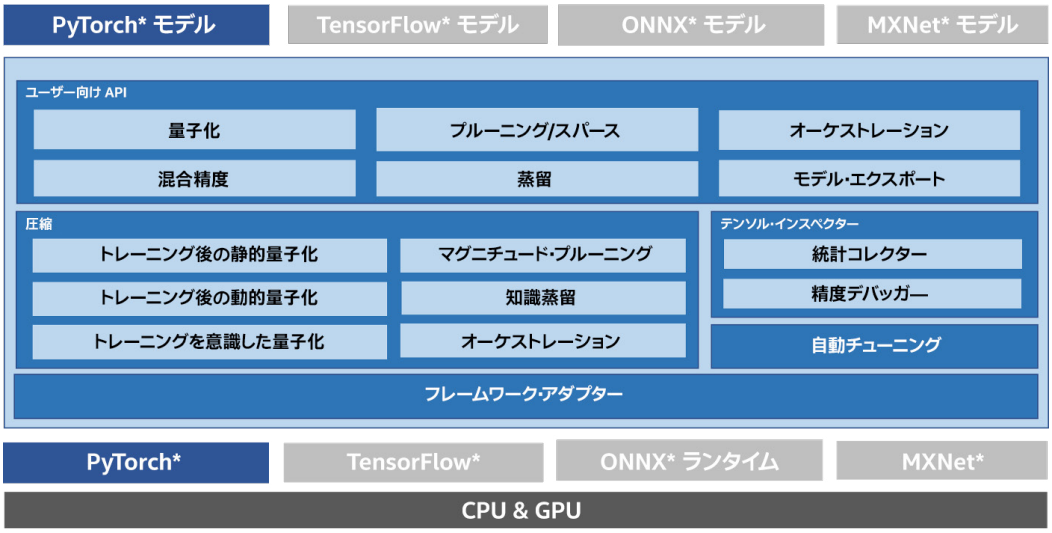


図 1. インテル® ニューラル・コンプレッサー

インテル® ニューラル・コンプレッサーは PyTorch* の量子化を拡張し、量子化と自動混合精度、精度を考慮したチューニング向けの高度なレシピを提供します。PyTorch* モデルを入力として受け取り、最適なモデルを生成します。量子化機能は、標準の PyTorch* 量子化 API をベースに構築されており、モデルレベルからオペレーター・レベルまでのきめ細かい量子化粒度をサポートするため、独自の変更を加えています。このアプローチにより、手動でチューニングをすることなく、より高い精度を得ることができます。

さらに、PyTorch* 自動混合精度機能を拡張し、第 3 世代インテル® Xeon® スケーラブル・プロセッサーでは、BF16 と FP32 に加えて INT8 をサポートしています。まず、量子化可能なオペレーターをすべて FP32 から INT8 に変換し、BF16 カーネルが PyTorch* でサポートされ、ベースとなるハードウェアで高速化されている場合は、残りの FP32 オペレーターを BF16 に変換します（**図 2**）。

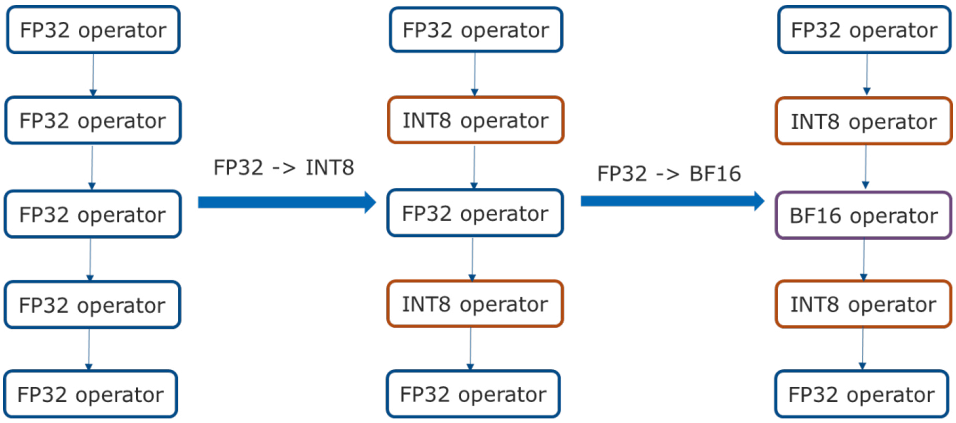


図 2. 自動混合精度

© Intel Corporation. Intel、インテル、Intel ロゴ、その他のインテルの名称やロゴは、Intel Corporation またはその子会社の商標です。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

インテル® ニューラル・コンプレッサーは、量子化の生産性を高めるため、精度を考慮した自動チューニング・メカニズムをサポートしています。まず、量子化粒度（per_tensor または per_channel）、量子化スキーム（対称または非対称）、量子化データ型（u8 または s8）、キャリブレーション手法（最小値 - 最大値または KL 発散）など、量子化機能についてフレームワークに問い合わせ（**図 3**）、次に各オペレーターでサポートされているデータ型を問い合わせます。結果に応じて、ツールは量子化設定の異なるセットからなるチューニング空間全体を生成し、チューニングの繰り返しを開始します。量子化設定の各セットについて、キャリブレーション、量子化、評価を行い、評価が目標精度に達すると、ツールはチューニング・プロセスを終了し、量子化モデルを生成します。

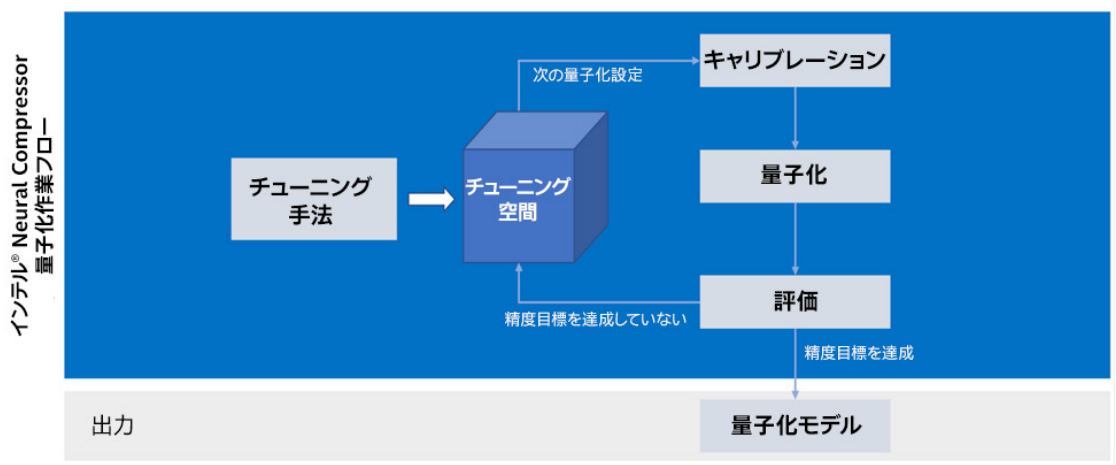


図 3. 精度を意識した自動チューニング

プルーニングでは、主に非構造化および構造化ウェイト・プルーニングとフィルター・プルーニングに注目します。非構造化プルーニングは、トレーニング中に重みの大きさが事前定義されたしきい値未満になった場合、マグニチュード・アルゴリズムを使用して重みを削除します。構造化プルーニングは、実験的なタイル単位のスパースカーネルを実装し、スパースモデルのパフォーマンスを向上させ、フィルター・プルーニングは、勾配によって計算された重要度スコアに応じて、モデル内のヘッド、中間層、隠れ状態を削除する勾配に基づくアルゴリズムを実装します。

インテル® ニューラル・コンプレッサーは、知識蒸留アルゴリズムも実装しており、大きな「教師」モデルから小さな「生徒」モデルへ、妥当性を損なうことなく知識を伝達します（**図 4**）。両モデルには同じ入力が与えられ、生徒モデルはその結果を教師および正しい結果ラベルの両方と比較しながら学習します。

© Intel Corporation. Intel、インテル、Intel ロゴ、その他のインテルの名称やロゴは、Intel Corporation またはその子会社の商標です。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

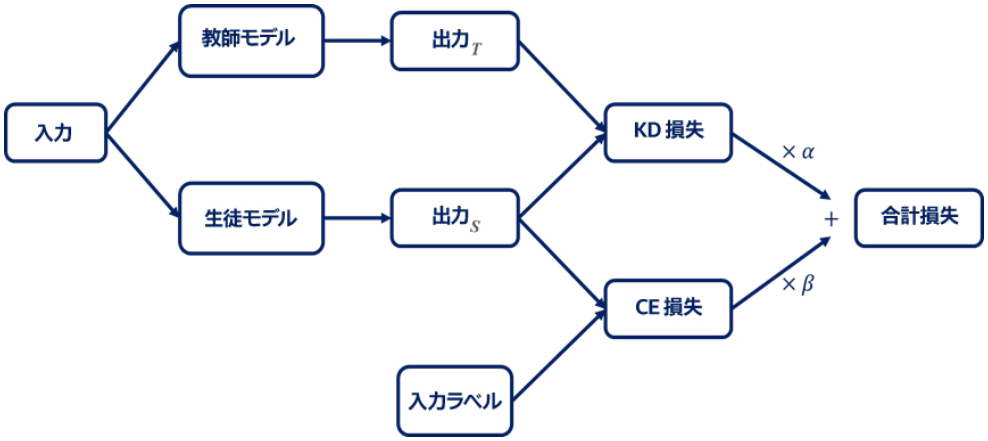


図 4. 知識蒸留アルゴリズム

次の例は、インテル® ニューラル・コンプレッサーを使用して自然言語処理モデルを量子化する方法を示しています。

```
# config.yaml
model:
  name: distilbert
  framework: pytorch_fx
tuning:
  accuracy_criterion:
    relative: 0.01
```

```
# main.py
import torch
import numpy as np
from transformers import (
    AutoModelForSequenceClassification,
    AutoTokenizer
)

model_name = "distilbert-base-uncased-finetuned-sst-2-english"
model = AutoModelForSequenceClassification.from_pretrained(
    model_name,
)
```

© Intel Corporation. Intel、インテル、Intel ロゴ、その他のインテルの名称やロゴは、Intel Corporation またはその子会社の商標です。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

```
# キャリブレーション・データローダー
class CalibDataLoader(object):
    def __init__(self):
        self.tokenizer = AutoTokenizer.from_pretrained(model_name)
        self.sequence = "Shanghai is a beautiful city!"
        self.encoded_input = self.tokenizer(
            self.sequence,
            return_tensors='pt'
        )
        self.label = 1 # negative sentence: 0; positive sentence: 1
        self.batch_size = 1

    def __iter__(self):
        yield self.encoded_input, self.label

# 評価関数
def eval_func(model):
    output = model(**calib_dataloader.encoded_input)
    print("Output: ", output.logits.detach().numpy())
    emotion_type = np.argmax(output.logits.detach().numpy())
    return 1 if emotion_type == calib_dataloader.label else 0

# 量子化
from neural_compressor.experimental import Quantization
quantizer = Quantization('./config.yaml')
quantizer.model = model
quantizer.calib_dataloader = CalibDataLoader()
quantizer.eval_func = eval_func
q_model = quantizer.fit()
```

生成される混合精度モデルは、低精度カーネルと使用するハードウェアによって異なります（例：第 3 世代インテル® Xeon® スケーラブル・プロセッサー上の INT8/BF16/FP32 混合精度モデル）。

パフォーマンス結果

インテル® ニューラル・コンプレッサーは、インテル® Xeon® Platinum 8380 プロセッサー上で 400 以上のサンプル（英語）を検証し、最小限の精度損失で 2.2 倍のパフォーマンス向上を達成しました（**表 1**）。検証されたモデルの詳細は、こちら（英語）を参照してください。

© Intel Corporation. Intel、インテル、Intel ロゴ、その他のインテルの名称やロゴは、Intel Corporation またはその子会社の商標です。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。

フレームワーク	モデル	精度			パフォーマンス/スループット (サンプル/秒)		
		INT8	FP32	精度比 [(INT8-FP32) /FP32]	INT8	FP32	パフォーマンス比 [INT8/FP32]
PyTorch* 1.11.0 + CPU	albert_base_mrpc	88.06%	88.50%	-0.50%	34.28	29.54	1.16x
	barthez_mrpc	82.99%	83.81%	-0.97%	166.84	89.56	1.86x
	bert_base_cola	58.80%	58.84%	-0.07%	260	126.47	2.06x
	bert_base_mrpc	90.28%	90.69%	-0.45%	251.79	126.46	1.99x
	bert_base_mrpc_qat	89.60%	89.50%	0.11%	258.89	125.79	2.06x
	bert_base_rte	69.31%	69.68%	-0.52%	252.14	126.45	1.99x
	bert_base_sst-2	91.97%	91.86%	0.12%	258.98	126.42	2.05x
	bert_base_sts-b	89.13%	89.75%	-0.68%	249.57	126.39	1.97x
	bert_large_cola	62.88%	62.57%	0.49%	88.75	36.7	2.42x
	bert_large_mrpc	89.93%	90.38%	-0.49%	89.43	36.62	2.44x
	bert_large_qnli	90.96%	91.82%	-0.94%	91.27	37	2.47x
	bert_large_rte	71.84%	72.56%	-1.00%	77.62	36.01	2.16x
	camembert_base_mrpc	86.56%	86.82%	-0.30%	241.39	124.77	1.93x
	deberta_mrpc	91.17%	90.91%	0.28%	152.09	85.13	1.79x
	distilbert_base_mrpc	88.66%	89.16%	-0.56%	415.09	246.9	1.68x
	distilbert_base_mrpc_fx	88.74%	89.16%	-0.47%	459.93	245.33	1.87x
	flaubert_mrpc	81.01%	80.19%	1.01%	644.05	457.32	1.41x
	inception_v3	69.43%	69.52%	-0.13%	454.3	213.7	2.13x
	longformer_mrpc	90.59%	91.46%	-0.95%	21.51	17.45	1.23x
	maskrcnn_fx	37.70%	37.80%	-0.26%	17.61	5.76	3.06x
	mbart_wnli	56.34%	56.34%	0.00%	65.05	31.26	2.08x
	peleenet	71.64%	72.10%	-0.64%	502.01	391.31	1.28x
	resnet18	69.57%	69.76%	-0.27%	800.43	381.27	2.10x
	resnet18_fx	69.57%	69.76%	-0.28%	811.09	389.36	2.08x
	resnet18_qat	69.74%	69.76%	-0.03%	804.76	388.67	2.07x
	resnet18_qat_fx	69.73%	69.76%	-0.04%	806.44	386.59	2.09x
	resnet50	75.98%	76.15%	-0.21%	507.55	200.52	2.53x
	resnet50_qat	76.04%	76.15%	-0.14%	490.64	203.49	2.41x
	resnext101_32x8d	79.08%	79.31%	-0.29%	203.54	73.85	2.76x
	rnnt	92.45	92.55	-0.10%	79.21	20.47	3.87x
	roberta_base_mrpc	87.88%	88.18%	-0.34%	250.21	124.92	2.00x
	se_resnext50_32x4d	78.98%	79.08%	-0.13%	358.63	173.03	2.07x
	squeezebert_mrpc	87.77%	87.65%	0.14%	249.89	207.43	1.20x
	transfo_xl_mrpc	81.97%	81.20%	0.94%	11.25	8.34	1.35x
	yolo_v3	24.60%	24.54%	0.21%	108.09	40.02	2.70x

表 1. インテル® ニューラル・コンプレッサーのパフォーマンス結果。システム構成：2022 年 6 月 10 日現在のインテル社内での測定値。プロセッサー:2x インテル® Xeon® Platinum 8380 プロセッサー @ 2.30GHz、40 コア /80 スレッド、インテル® ターボ・ブースト・テクノロジー有効、インテル® ハイパースレッディング・テクノロジー有効。メモリー：256GB（16x16GB DDR4 3200MT/秒）。ストレージ：インテル® SSD x1。NIC：2x イーサネット・コントローラー 10G X550T。BIOS：SE5C6200.86B.0022.D64.2105220049（ucode:0xd0002b1）。OS：Ubuntu* 20.04.1 LTS。カーネル：5.4.0–42-generic。バッチサイズ：1。インスタンスごとのコア数：4。

インテル® ニューラル・コンプレッサー（英語）は、一般的なニューラル・ネットワーク圧縮アプローチに適用可能な自動チューニング機構と使いやすい API により、生産性を向上し、精度の損失を低減することを目的としています。インテルでは、より多くの圧縮レシピを追加し、それらを組み合わせて最適なモデルを生成することで、このツールを継続的に改善しています。ぜひ、インテル® ニューラル・コンプレッサーをお試しいただき、GitHub* リポジトリー（英語）からフィードバックを提供してください。

© Intel Corporation. Intel、インテル、Intel ロゴ、その他のインテルの名称やロゴは、Intel Corporation またはその子会社の商標です。
コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。