深層学習の原理探究へ向けたRust製フレームワークの構築

GitHub リポジトリ

逗子開成高等学校　高校2年　臼井千裕　鈴木翔天

SCIENCE

SINCE 1903

研究概要

Rustは近年注目されている言語である一方、**機械学習**の分野においては開発途上でありpythonやC系言語に比べて**文献が少ない**。

深層学習の分野ではフレームワークのコミュニティやドキュメントの多くが**英語**によって説明されているため学習や問題解決までの障壁が高い。

実用的でフルRust実装のフレームワーク

日本語によってコードの説明がなされていてユーザー自身の手で実装してもらう

深層学習の原理探究に向けたRust製フレームワークを構築

我々は開発したフレームワークを***StuCrs***と名付ける

研究にあたって

本研究は下の著書『**ゼロから作るDeep Learning③　フレームワーク編**』をもとにして実装しています。著者である**斎藤康毅**氏に著書の考えや表現の使用を許可していただいたことに感謝を申し上げるとともに、この著書オリジナルのフレームワーク**DeZero**も研究の参考として利用させていただいています。

DeZero

*Rust*とは

Rustの特徴は**実行速度の速さ**、**安全性の担保**、**モダンな言語**が挙げられます。

実行速度の速さ
- 直接コンパイル
- ガベージコレクションがない
- ゼロコスト抽象化

最も実行速度が速いといわれるC,C++に匹敵する

安全性の担保

メモリー ← 監視 ← コンパイラー

未然にメモリ安全やスレッド安全ではない操作を防ぐ

モダンな言語
- 強力な型推論
- 不変、可変
- トレイト
- 代数的データ型

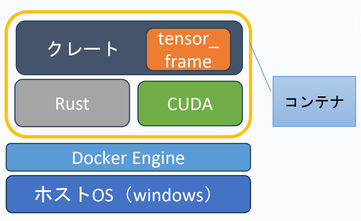
様々なプログラミング言語の機能が入っている

環境開発

Dockerを用いて**Rust**の環境をそろえたコンテナを**Visual Studio Code**で立ち上げる。

NVIDIA製のGPUを用いて学習できるようNVIDIAの**CUDA、tensor_frame**をコンテナ内にインストールする。

＜環境イメージ＞

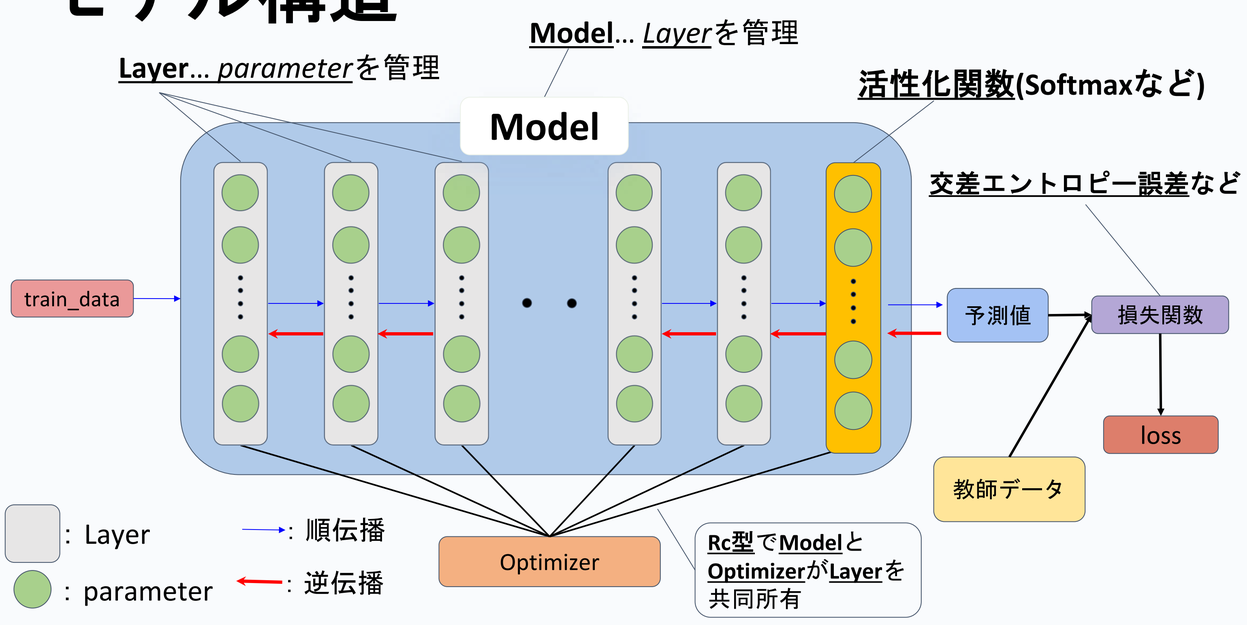


- **Docker…**コンテナ型の仮想環境を構成、管理するソフトウェア。
- **CUDA…** NVIDIAが提供するGPU向けの統合開発環境。
- **ndarray…**多次元配列を効率的に扱うためのクレート。
- **Tensor_frame…** RustからGPU上で行列計算を扱うためのクレート。
  - ➡ **ndarray**のように高度な行列計算を行える。

※詳しいクレートのバージョンや依存関係、スペックはGitHubリポジトリをご参照ください。

<スペック>
- CPU: intel Core i5 12400,
- GPU: GeForce RTX 4060,8GB
- メモリ: 48GB

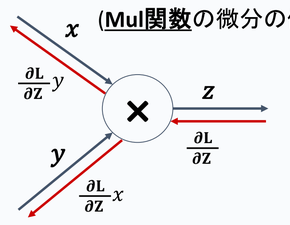
モデル構造



実装への主な流れ

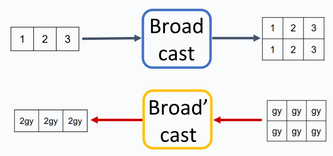
実数での自動微分
1. ***RcVariable***, ***Function トレイト***を実装
2. 一変数関数の微分 (sin, log, tanhなど)
3. 複雑な関数の微分を自動化

(**Mul関数**の微分の例)



演算子のオーバーロード
1. 演算子の関数を実装(Add,Mul,Subなど)
2. 演算子を***RcVariable***の演算子にオーバーロード
   add(&a,&b) → **&a + &b** と書くことができる。

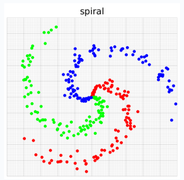
(**Broadcast関数**の逆伝播の例)



行列への対応
1. f32型を**ndarray型**に変更
2. 行列関数を実装(reshape, sumなど)
3. ブロードキャストへの対応

ニューラルネットワークの構築
1. ***Layer***, ***Model***, ***Optimizer***を実装
2. **損失関数**を実装(**交差エントロピー誤差**など)
3. ***Dataset***, ***DataLoader***を実装
4. 自作の**スパイラルデータ**で多値分類を試す

自作したスパイラルデータを図にプロット



MNISTの学習
1. **MNIST**のデータセットをダウンロード、実装
2. 学習、および推論
3. 他の**フレームワーク**と比較

TensorFlow

DeZero

公開の準備
1. パッケージの作成
2. リポジトリの公開
3. ドキュメントの作成

※詳しい実装までの説明はGitHubリポジトリのドキュメントをご参照ください。

*GPU*対応

ここではRustの**tensor_frame**というクレートを用いる。
このクレートは**ndarray**と同じように、**ブロードキャスト**と行列計算に対応し、かつGPU上で計算することができる。

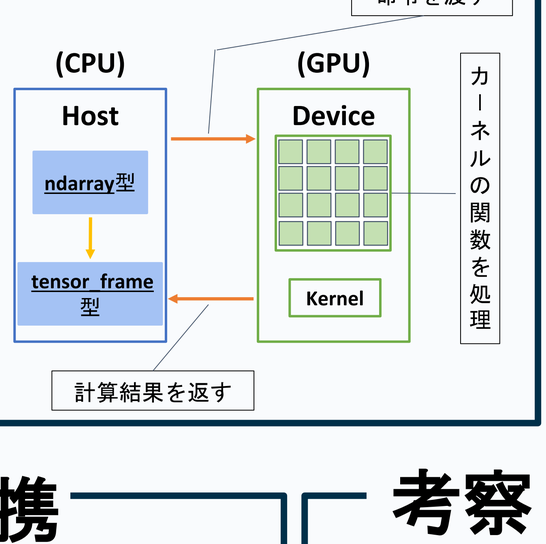
GPU用のコードをもう一つ用意

↓

ndarrayから**tensor_frame**の行列構造体に変換できるよう設定

↓

各関数でGPU上で計算できるよう変更



MNIST での性能比較

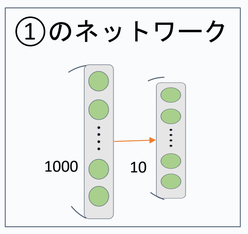
※詳しい学習のデータはリポジトリに載せてあります。

下の表は3種類のフレームワーク(**StuCrs**,**Dezero**,**Tensorflow**)で**MNIST**のデータをCPUで学習した際の処理にかかった時間を表している。
今回は二種類のニューラルネットワークは構築し、比較した。

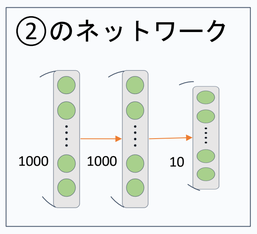
学習の設定
- epoch… 5
- batch… 100

①… 出力が(1000,10)、活性化関数がReLUのDenseレイヤ
②… 出力が(1000,1000,10)、活性化関数がReLUのDenseレイヤ

①のネットワーク



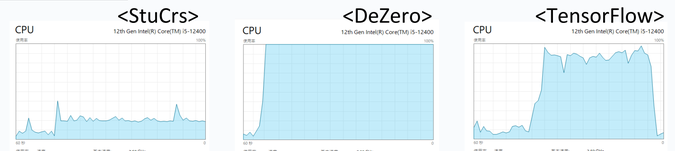
②のネットワーク



また②における学習でそれぞれのCPU使用率を計測した。
右下のグラフが使用率を表しており、これを見ると、
StuCrsが他二つに比べて**CPU使用率が低く抑えられている。**

フレームワーク	StuCrs	DeZero	TensorFlow
①のネットワーク	20.99382s	35.70813s	17.60617s
②のネットワーク	48.97416s	76.68909s	35.39685s

<StuCrs>　<DeZero>　<TensorFlow>



*GitHub*との連携

本研究は**GitHubリポジトリ**として公開している。
また、リポジトリでは**自ら作成したドキュメント**でプログラミングの概要を説明している。

・ドキュメント

日本語でプログラミングを詳細に説明することによって、ユーザーが**直観的に理解しやすい**ような設計になっている。

深層学習の枠組みがわかりやすく説明されているため、ユーザー自身の手で実装してもらうことができ、**深層学習の原理探求につながる**。

・リポジトリの公開

リポジトリを公開することで様々な方からアドバイスや改善点を教えてもらうことができ、より良い品質のコードが作れるようになる。

考察

今回構築したフレームワークにおいて**「MNISTでの性能比較」**、**「GitHubとの連携」**から次のようなことがいえる。

性能面
CPUでの学習においては層が浅いとTensorflowにせまる性能を持ち、DeZeroと比べて**約2倍近くの性能**があるため、実用的であると言える。

機能面
日本人にとって、**とても分かりやすい設計**になっており、ドキュメントを見ることでなぜそのプログラミングを書くのかが理解できる。

改善点
DataLoaderはデータを複製するため**実行速度とメモリに余計なコストがかかってしまう**。また、行列データが小さいほど、StuCrsはDezeroと比べてより高速に処理するため、行列計算を担う**ndarrayクレート**の処理が**numpy**などよりも遅いことがわかる。さらに高速な行列計算クレートが開発されれば、さらなる**フレームワークの高速化が期待できる**。

反省 ,今後の展望

今回の研究ではRustを用いて実際に１からフレームワークを構築することで**深層学習の仕組みを理解することができた**。
しかしTensorFlowなどの既存のフレームワークと比べるとプログラムの最適化ができていないため、実行速度としてはまだ劣っており、機能面においてもまだまだ**実装できていない関数も多い**。

バグの対処やコメントによる説明を増やし、**コミュニティを拡大する**ことでユーザーの理解を深める。

OptimizerやCNN、LSTM用のレイヤなど、実装する関数の種類を増やす。

現状のフレームワークは型や行列の次元を指定しない**動的な関数**と指定する**静的な関数**が混ざっており、まとまりがない状態なのでユーザーが理解しやすいように整理、体系化する。

参考文献
- 『ゼロから作るDeep Learning③　フレームワーク編』　斎藤康毅　(2020) オライリー・ジャパン
- 『実践Rust入門[言語仕様から開発手法まで]』keen、 河野達也、小松礼人　(2019) 技術評論社
- 『pythonで動かして学ぶ！あたらしい深層学習の教科書：　機械学習の基本から深層学習まで』石川聡彦 (2018) 翔泳社
- 『python3年生ディープラーニングの仕組み体験してわかる会話でまなべる！』巧尚　(2023) 翔泳社
- 『実践RUSTプログラミング入門』　初田直哉、山口聖弘、吉川智史、豊田優貴、松本健太郎、原将己、中本謙弘　(2020) 株式会社秀和システム
- 『やさしいRust入門』日向俊二　(2021) カットシステム
- 『AlphaZero 深層学習・強化学習・探索 人工知能プログラミング実践入門』布留川英一　(2019) ボーンデジタル

使用した主なクレート
- Tensor_frame <https://docs.rs/tensor_frame/latest/tensor_frame/index.html>
- ndarray <https://docs.rs/ndarray/0.16.0/ndarray/index.html>
- plotters <https://docs.rs/plotters/0.3.1/plotters/index.html>
- mnist <https://docs.rs/mnist/latest/mnist/index.html>
- image <https://docs.rs/image/latest/image/index.html>

使用したロゴ
- Tensorflow <https://www.tensorflow.org/?hl=ja>
- PyTorch <https://pytorch.org/>
- NumPy <https://numpy.org/ja/>
- python <https://www.python.org/>
- GitHub <https://github.com/>
- Rust <https://www.rust-lang.org/ja>

※詳細な参考文献リストは、GitHubのリポジトリをご参照ください。