






文書管理ソフトウェアを作った話

Akira Kawata

<https://akawashiro.github.io/>

2023/02/16 KMC 例会講座

jendeley <https://github.com/akawashiro/jendeley>

REGISTER WEBPAGE			REGISTER PDF FROM URL			UPLOAD PDF			SEARCH		
title	authors	tags	comments	y...	↓	⋮					
Filter by title	Filter by authors	Filter by tags	Filter by comments	year							
 Modular Formal Verification of Rust Programs with Unsafe Blocks	Nima Rahimi Foroushaani Bart Jacobs	hoge	hohogefuga							2022	
 Benchmarking Large Language Models for Automated Verilog RTL Code Generation	Shailja Thakur Baleegh Ahmad Zhenxing Fan Hammond Pearce Benjamin Tan Ramesh Karri Brendan Dolan-Gavitt Siddharth Garg	2023-01-25								2022	
 RustHorn: CHC-based Verification for Rust Programs (full version)	Yusuke Matsushita Takeshi Tsukada Naoki Kobayashi									2020	

プログラミングに関する文書が多様すぎる

- プログラミング言語のリファレンス
- ソフトウェアのソースコード
- ハードウェアの仕様書
- Stack Overflow
- 論文
- 本

cppreference.com [Create account](#)

Page Discussion [View](#) [Edit](#) [History](#)

C++ Utilities library

std::move

Defined in header <utility>

```
template<class T>
typename std::remove_reference<T>::type&& move( T&& t ) noexcept; (since C++11)
template<class T>
constexpr std::remove_reference<T>&& move( T&& t ) noexcept; (since C++14)
```

std::move is used to *indicate* that an object *t* may be "moved from", i.e. allowing the efficient transfer of resources from *t* to another object.

In particular, std::move produces an *rvalue expression* that identifies its argument *t*. It is exactly equivalent to a *static_cast* to an *rvalue reference* type.

Parameters

t - the object to be moved

Return value

```
static_cast<typename std::remove_reference<T>::type&&>(t)
```

Notes

The functions that accept *rvalue reference* parameters (including move constructors, move assignment operators, and regular member functions such as `std::vector::push_back`) are selected, by overload resolution, when called with *rvalue arguments* (either *rvalues* such as a temporary object or *rvalues* such as the one produced by `std::move`), if the argument identifies a resource-owning object, these overloads have the option, but aren't required, to move any resources held by the argument. For example, a move constructor of a linked list might copy the pointer to the head of the list and store `nullptr` in the argument instead of allocating and copying individual nodes.

Names of *rvalue reference* variables are *rvalues* and have to be converted to *rvalues* to be bound to the function overloads that accept *rvalue reference* parameters, which is why move constructors and move assignment operators typically use `std::move`:

```
// Simple move constructor
A(A&& arg) : member(std::move(arg.member)) // the expression "arg.member" is lvalue
{}
// Simple move assignment operator
A& operator=(A&& other) {
    member = std::move(other.member);
    return *this;
}
```

One exception is when the type of the function parameter is *rvalue reference* to type template parameter ("forwarding reference" or "universal reference"), in which case `std::forward` is used instead.

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{he, xiangyu, ren, jiasun}@microsoft.com

Abstract

Deep residual networks are now difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning conventional functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we achieve residual error rates with a depth of up to 120 layers that are comparable to baseline models with depth of up to 18 layers. A deeper than VGG on ILSVRC14, but still having lower complexity. An accuracy of 68.4% is achieved on ImageNet 2015 on the ImageNet2015 classification task. We also present an analysis of the training process and the effect of different residual architectures. The depth of architectures is of central importance for many visual recognition tasks. Study that use a network deep approximation, we obtain a 29% relative improvement on the COCO object detection dataset. Deep residual are an extension of our submission to ILSVRC14 COCO 2015 competition, where we also won the 1st place on the task of ImageNet detection. ImageNet localization COCO detection, and COCO segmentation.

Figure 1. Training error (left) and test error (right) on ILSVRC14 with 18-layer and 34-layer "bottleneck" networks. The deeper network has higher training error, and this is not error. Similar phenomena are observed in Figure 2.

Figure 2. Training error (left) and test error (right) on COCO10 with 18-layer and 34-layer "bottleneck" networks. The deeper network has higher training error, and this is not error. Similar phenomena are observed in Figure 3.

When deeper networks are able to start overfitting, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be interpreted as overfitting) and then degrades rapidly. Unexpectedly, such degradation is not caused by overfitting, and adding more layers to a scalable deep model leads to higher training error, as reported in [11, 41] and thoroughly verified by our experiments. Fig. 1 shows typical example.

The degradation of training accuracy indicates that not all features are equally easy to represent. Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. They consist a sublayer in common, while the deeper model the added layers are located respecting, and the other layers are copied from the learned shallower model. The existence of this contextual relation indicates that a deeper model should produce a higher training error than its shallower counterpart. But experiments show that our current solution is based on model to find solutions that

FLEMMING NIELSEN
HANNEFRIS NIELSEN
CHRIS HANKIN

Principles of Program Analysis

Springer

文書を整理したい

- 多様な形式の文書を扱いたい
- タグを使って分類したい
- コメントを書きたい
- 検索、情報処理したい
 - データベースがテキストファイルであってほしい
- 長期的に保存したい
 - 10年単位で保存したい
 - ローカルで動いてほしい
 - ウェブサービスは不可
 - 自分で保守したい
- クロスプラットフォームで動いてほしい







既存のソフトウェア

- Zotero, Paperpile, Mendeley
 - 多様な形式の文書 => 🧑
 - データベースがテキストファイル => 🧑
 - ローカルでの動作 => 🧑
 - 自分で保守 => 🧑
- JabRef
 - 多様な形式の文書 => 🧑
 - データベースがテキストファイル => ?
 - ローカルでの動作 => 🧑
 - 自分で保守 => ?

zotero



jendeleyを作りました!

	REGISTER WEBPAGE	REGISTER PDF FROM URL	UPLOAD PDF					
	title	authors	tags	comments	y...			
	Filter by title	Filter by authors	Filter by tags	Filter by comments	year			
	Modular Formal Verification of Rust Programs with Unsafe Blocks	Nima Rahimi Foroushaani Bart Jacobs	hoge	hogehogefuga	2022			
	Benchmarking Large Language Models for Automated Verilog RTL Code Generation	Shailja Thakur Baleegh Ahmad Zhenxing Fan Hammond Pearce Benjamin Tan Ramesh Karri Brendan Dolan-Gavitt Siddharth Garg	2023-01-25		2022			
	RustHorn: CHC-based Verification for Rust Programs (full version)	Yusuke Matsushita Takeshi Tsukada Naoki Kobayashi			2020			

使ってみてください

```
npm install @a_kawashiro/jendeley -g  
jendeley scan --papers_dir <YOUR PDFs DIR>  
jendeley launch --db <YOUR PDFs DIR>/jendeley_db.json
```

- npm: https://www.npmjs.com/package/@a_kawashiro/jendeley
- Github: <https://github.com/akawashiro/jendeley>
- ドキュメント: <https://akawashiro.github.io/jendeley/>
- 紹介記事 https://zenn.dev/a_kawashiro/articles/a2170f967f9508
- Hacker News: <https://news.ycombinator.com/item?id=34747285>

テモ

データベースの中身

```
> cat ~/Dropbox/papers/jendeley_db.json | head -n 15
{
  "jendeley_meta": {
    "idType": "meta",
    "version": "0.0.18"
  },
  "doi_10.1145/1122445.1122456": {
    "path": [
      "A Comprehensive Survey of Neural Architecture Search.pdf"
    ],
    "idType": "doi",
    "tags": [],
    "comments": "",
    "dataFromCrossref": {
      "indexed": {
        "date-parts": [
```

jendeleyの機能

- 多様な形式の文書 => 🧑 PDFもしくはウェブページなら何でもOK
- データベースがテキストファイル => 🧑 JSON
- ローカルでの動作 => 🧑 完全ローカルで動作
- 自分で保守 => 🧑 僕が作りました
- クロスプラットフォーム => 🧑 Windows/Linuxで毎日使っています
 - Dropboxでデータベースを同期

jendeleyの実装

ウェブサービスをローカルで動かすような作りになっている

- Frontend
 - React + Material React Table + MUI
- Backend
 - Express

お願い

- <https://github.com/akawashiro/jendeley> にスターしてください

