COMP5511 Artificial Intelligence Concepts

Comparisons of Neural Network Frameworks

Group D

The Hong Kong Polytechnic University

Table of contents

Introduction

Background

Nowadays, the deep neural network models, also called Deep Learning, have already became a significant machine learning methods.

The difficulty of choosing a neural network package may come from hardware level. Besides GPU, there are other different type of accelerators, like FPGA or ASIC(TPU). The different accelerators may limit the choice of neural network frameworks. Other hardware architectures like ARM or RISC-V would also restrict the usage of neural network frameworks.

It is necessary to evaluate them fairly and summarize their pros and cons before making the decision.

Datasets

We use two datasets, i.e. Szeged Weather dataset and MNIST dataset [1].

Column Id	Attribute Name	Domain
1	Formatted Date	Text
2	Summary	Text
3	Precip Type	Text
4	Temperature (C)	Number
5	Apparent Temperature (C)	Number
6	Humidity	Number
7	Wind Speed (km/h)	Number
8	Wind Bearing (degrees)	Number
9	Visibility (km)	Number
10	Loud Cover	Number
11	Pressure (millibars)	Number
12	Daily Summary	Text

Table 1: Szeged Weather dataset

Figure 1: MNIST dataset

Frameworks

FRAMEWORKS AND MODELS

Frameworks

- TensorFlow
- Keras
- Scikit-learn
- PyTorch

Models

- CNN
- MLP
- MLP.Regression

TensorFlow

TensorFlow is an open source library, which is developed by Google, for implementing machine learning. It can support multiple languages like python, java, JavaScript, Go and C etc.

Why TensorFlow?

- TensorFlow helps to ease the process of training models, acquiring data, serving predictions and refining future results
- A few lines codes to build a learning model
- TensorFlow makes machine learning and deep learning models and algorithms useful by way of a common metaphor through bundles them together

Keras

Keras is an easy-to-learn high-level Python library based on TensorFlow (or Theano or CNTK) for deep learning. It allows artificial intelligence engineers to focus on the main functions of deep learning.

Why Keras?

- Efficiently executing low-level tensor operations on CPU, GPU.
- Can be programmed based on different deep learning backends, such as TensorFlow, CNTK and Theano.
- Compared with other frameworks, Keras can be widely deployed on a wider range of platforms.

Scikit-learn

Scikit-learn is integrating a wide range of machine learning algorithms for dealing with medium-size supervised and unsupervised problems based on Python module. It was started as a Google Summer of code project by David Cournapeau in 2007.

Why Scikit-learn?

- Scikit-learn to give more machine learning function's API or achieved classes model to non-specialists.
- Scikit-learn package can provide most normal and popular algorithms to implement the machine learning such as SVM, KNN, SVR, etc.
- Scikit-learn is built on Numpy, Scipy, and Matplotlib.

PyTorch

PyTorch is an open-source machine learning library based on Torch library, used for applications such as computer vision and natural language processing. The predecessor of PyTorch is Torch which is a pure C++ framework. The PyTorch is mainly maintained by Facebook's AI Research Lab(FAIR), which ensures the PyTorch library can get continuously support and improvement.

Why PyTorch?

- Easily to get start
- PyTorch is a quite simple and efficient framework for neural network
- Its design obeys human thought, which makes users can focus on their ideas rather than the details of implementation

CNN

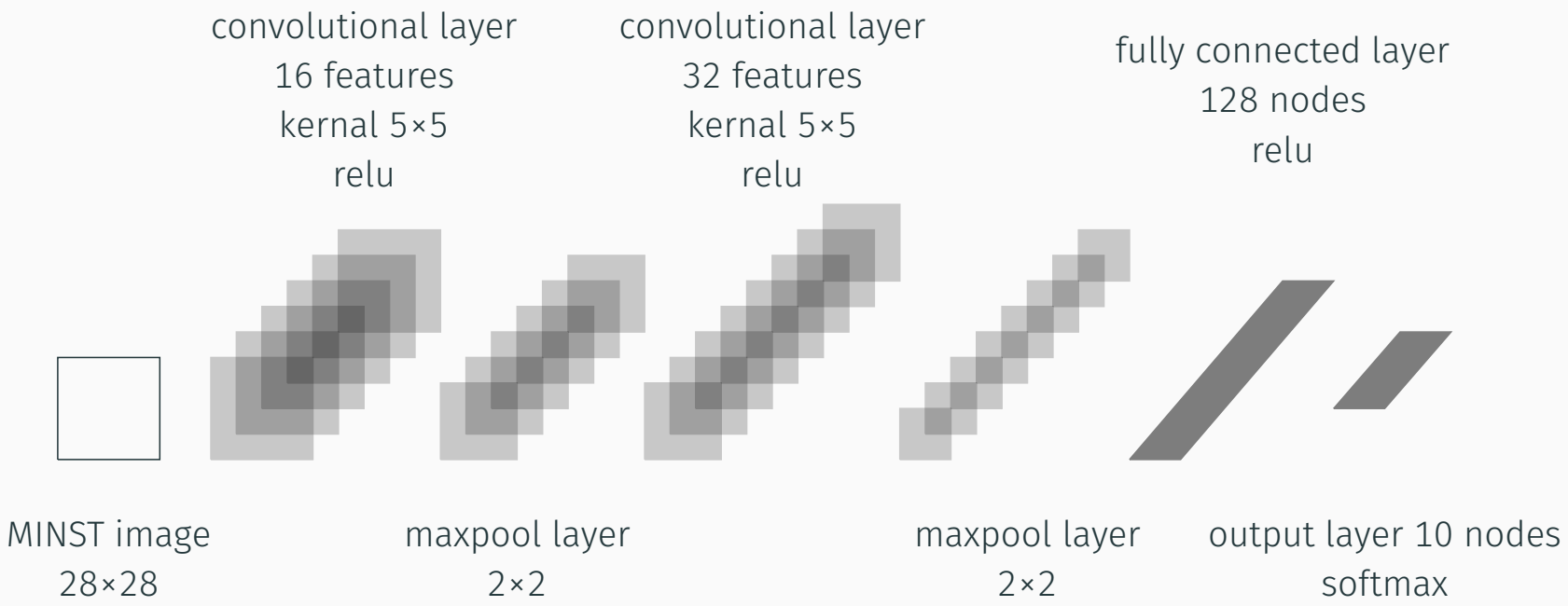

Figure 2: CNN with crossentropy loss function and adam optimizer for MNIST dataset

CNN [2] with two convolutional layers, and both of them follow by a max pooling layer. The first fully connective layer owns 128 units, and the output layer has 10 units.

MLP

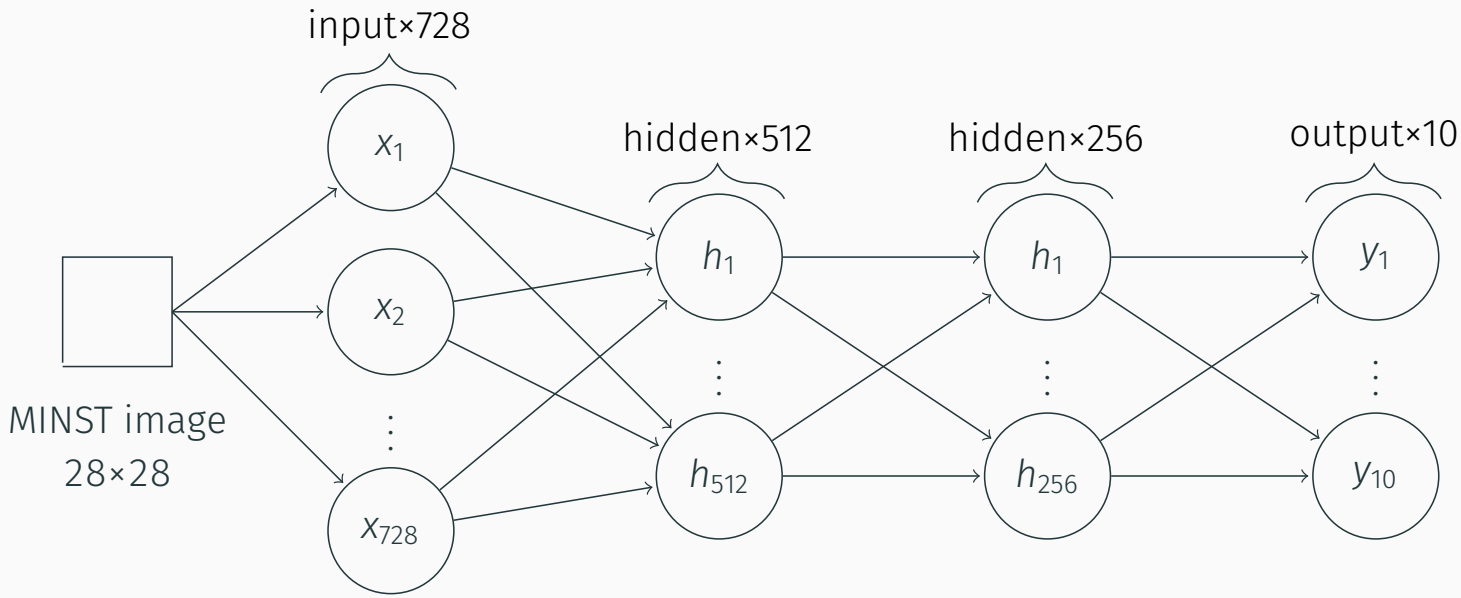

Figure 3: MLP with crossentropy loss function and adam optimizer for MNIST dataset

In the experiment, the MLP [3] models is a 3 layers neural network model, where the amount of units for input, the first hidden layer, the second hidden layer and the output layer are 728, 512, 256 and 10 separately.

MLP.Regression

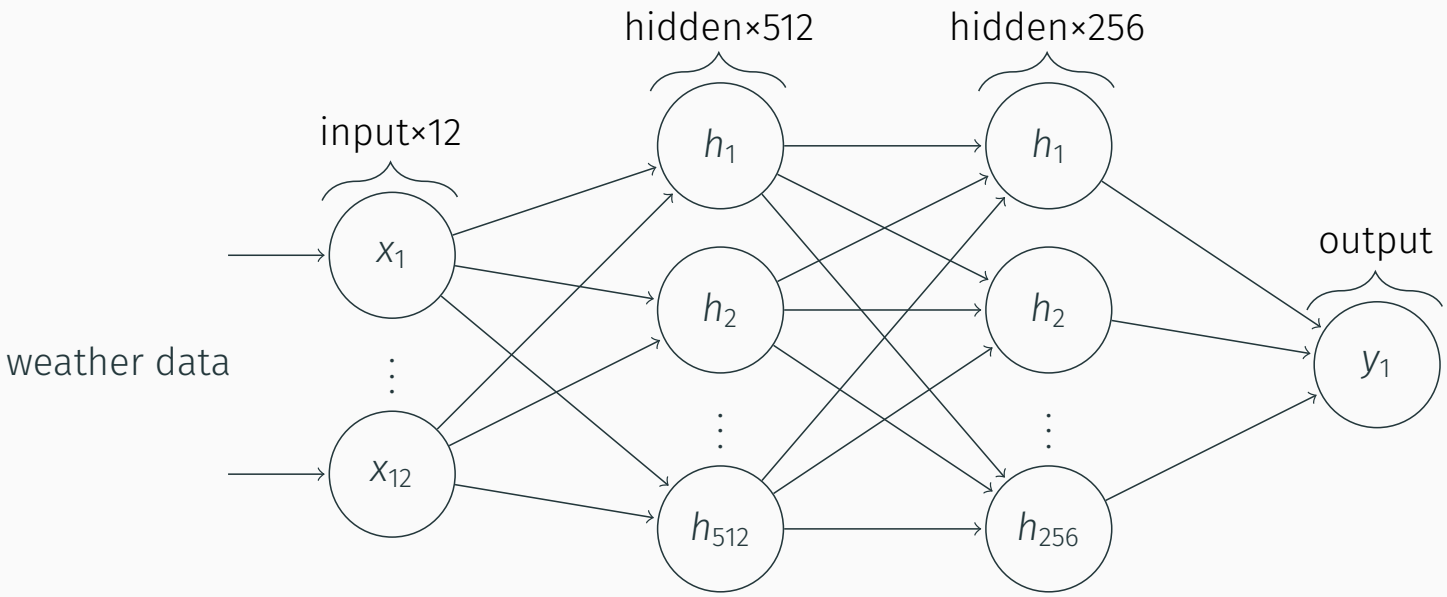

Figure 4: MLP Regression with MSE loss function and adam optimizer for MNIST dataset

Similar to the MLP model in the MNIST classification task.

Compared with the MLP model in the classification task, the input layer and output layer only have 12 and 1 units separately.

Comparisons

Comparison rubrics

- Performance
- Weight Initialization
- Activation/Loss function Coverage
- Optimization methods Coverage
- Hardware/Platform support
- Language support
- Usability
- Activity of Packages

Performance on CPU

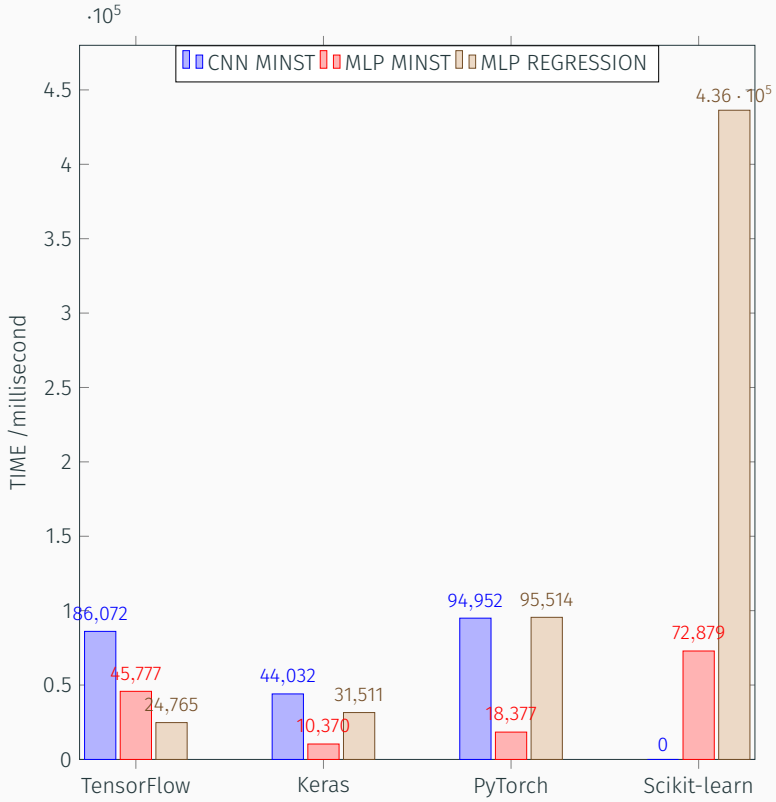

Figure 5: Training time on CPU

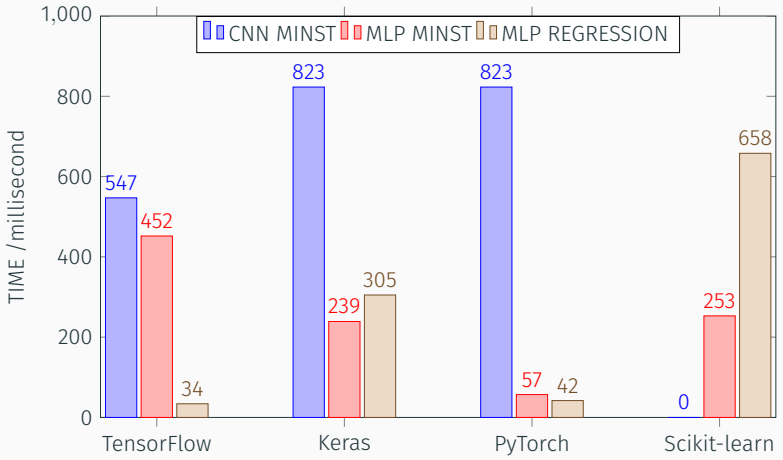

Figure 6: Prediction time on CPU

Keras has the shortest training time with relatively highest prediction time. Notably, Scikit-learn doesn't support CNN. The performance of PyTorch is between Keras and TensorFlow.

Performance on GPU

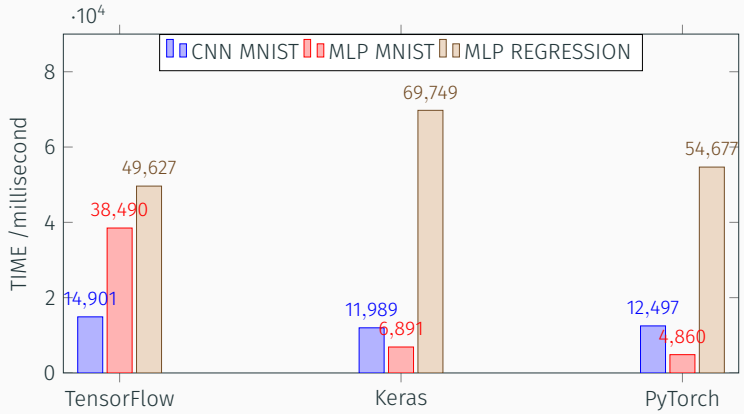

Figure 7: Training time on GPU(Scikit-learn cannot support GPU)

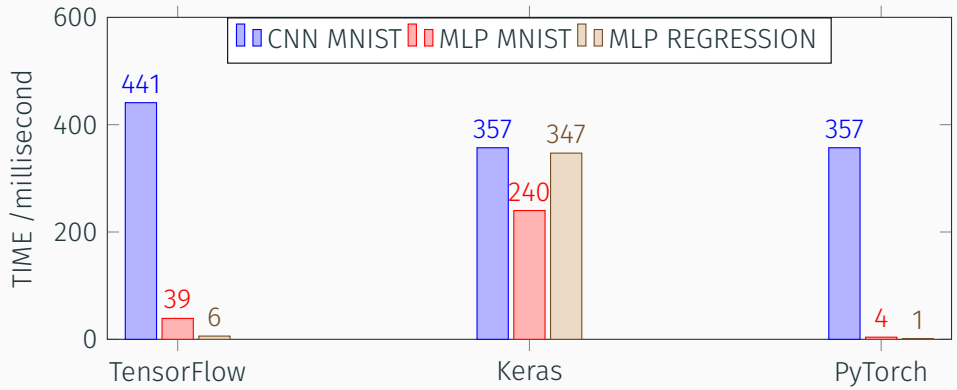

Figure 8: Prediction time on GPU

Within the same data and neural network model, the performance of different packages show on above the figures.

TensorFlow and Keras have the balanced good performance in training time on CPU and GPU.

Scikit-learn has the worst time complexity among four packages.

Weight Initialization

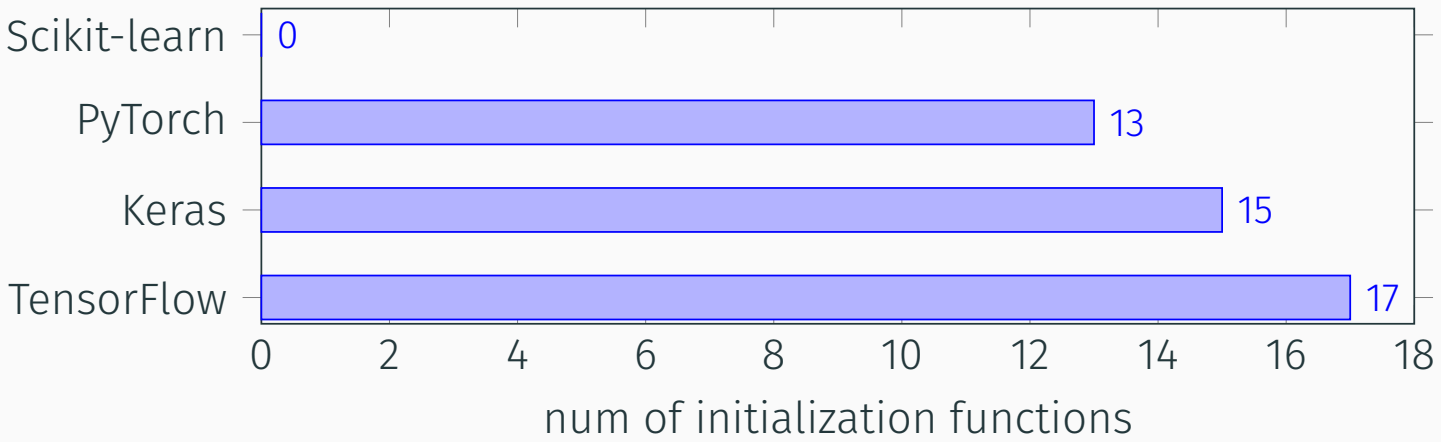

Figure 9: Number of initialization functions

In particular, Scikit-learn does not provide any initialization method, it initial the weights randomly.

Activation/Loss function Coverage and Optimization methods Coverage

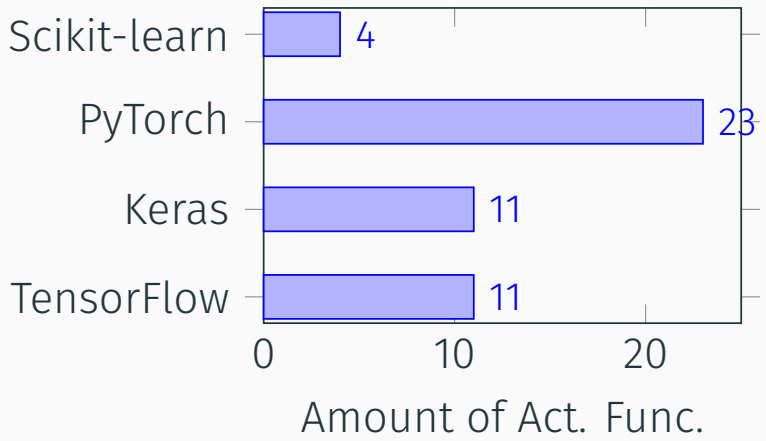

Figure 10: Activation/Loss function Coverage

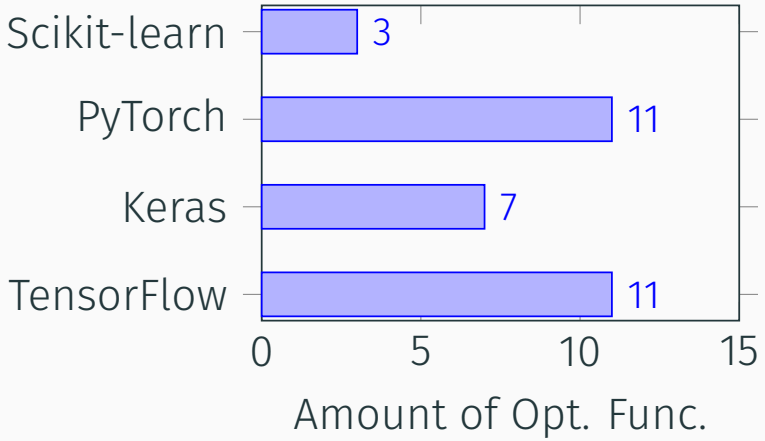

Figure 11: Optimization methods Coverage

PyTorch has 23 different activation/loss functions, so the PyTorch has better diversity to build neural network.

And it also has 11 different optimization methods, that means it also has wonderful ability of converging data.

Hardware support

Frameworks	AArch64/ARM	ppc64le/Power Architecture(IBM)	x86
TensorFlow	Yes	Yes	Yes
Keras	Yes	Yes	Yes
Scikit-learn	Yes (official support)	Yes (official support)	Yes (official support)
PyTorch	Yes (official support)	Yes (official support)	Yes (official support)

Table 2: Hardware support status

Frameworks	RISC.V	ASIC	FPGA	GPU
TensorFlow	Yes	Yes	No	Yes
Keras	Yes	Yes	No	Yes
Scikit-learn	Yes (LAPAC and gfortran are needed)	No	Yes	No
PyTorch	Yes (Recompile with LLVM > 9.0)	Exist solution	Exist solution	Yes (CUDA)

Table 3: Hardware support status Cont.

All the packages are satisfied running on CPU. Scikit-learn does not support ASIC and GPU. TensorFlow and Keras do not support FPGA.

Platform/Language support

Platform	Hadoop	Spark
TensorFlow	✓	✓
Keras	✓	✓
PyTorch	✓	✓
Scikit-learn	✓	

Table 4: platform support

Library	Python	Java	C	Go	Js	R	C++
TensorFlow	✓	✓	✓	✓	✓		
Keras	✓					✓	
PyTorch	✓	✓					✓
Scikit-learn	✓						

Table 5: language support

To sum up, TensorFlow is best in the Language support part, We can use 5 kind of language to call it.

Scikit-learn is worst, can only used by python.

Besides python, Keras can use R, PyTorch and use C++ and Java.

Usability

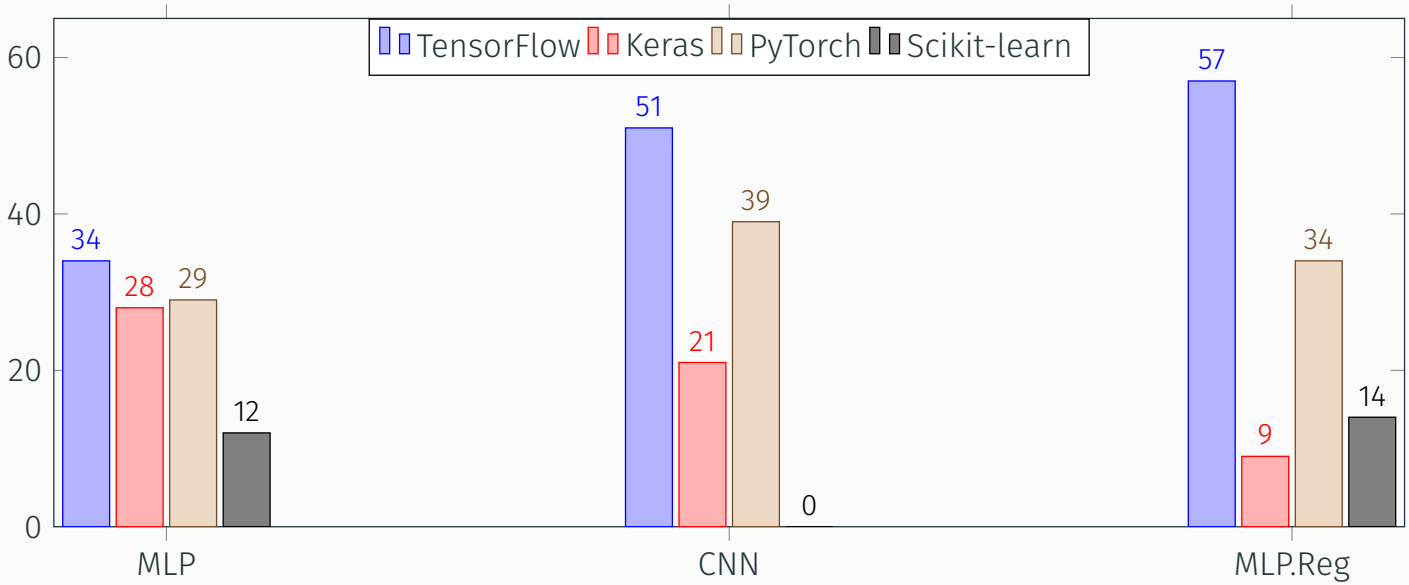

Figure 12: Code lines for construct the same structure neural network for four packages

From the picture, Scikit-learn has the less amount of code lines, that means it more friendly to novices.

Novices will meet more questions when they use TensorFlow at the beginning of study.

Activity of Packages

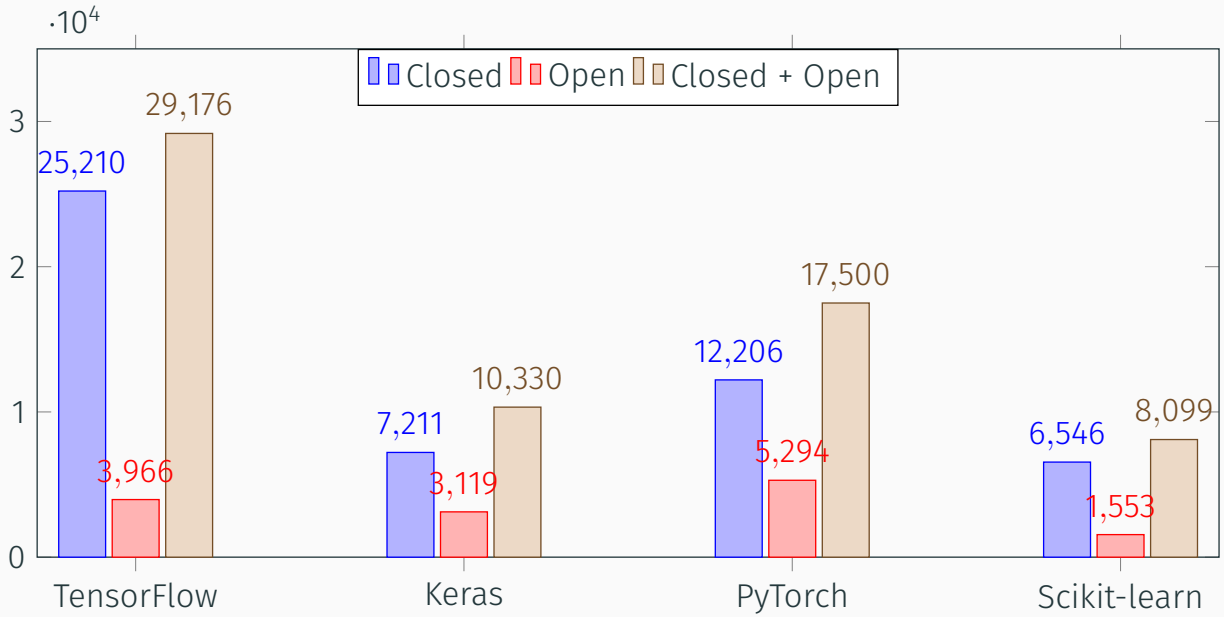

Figure 13: Total amount of open and closed issues for four packages (up to Dec, 2020)

Total amount of open and closed issues represents a package's sustainability and usability.

TensorFlow and PyTorch can provide more solutions to novices.

Activity of Packages Cont.

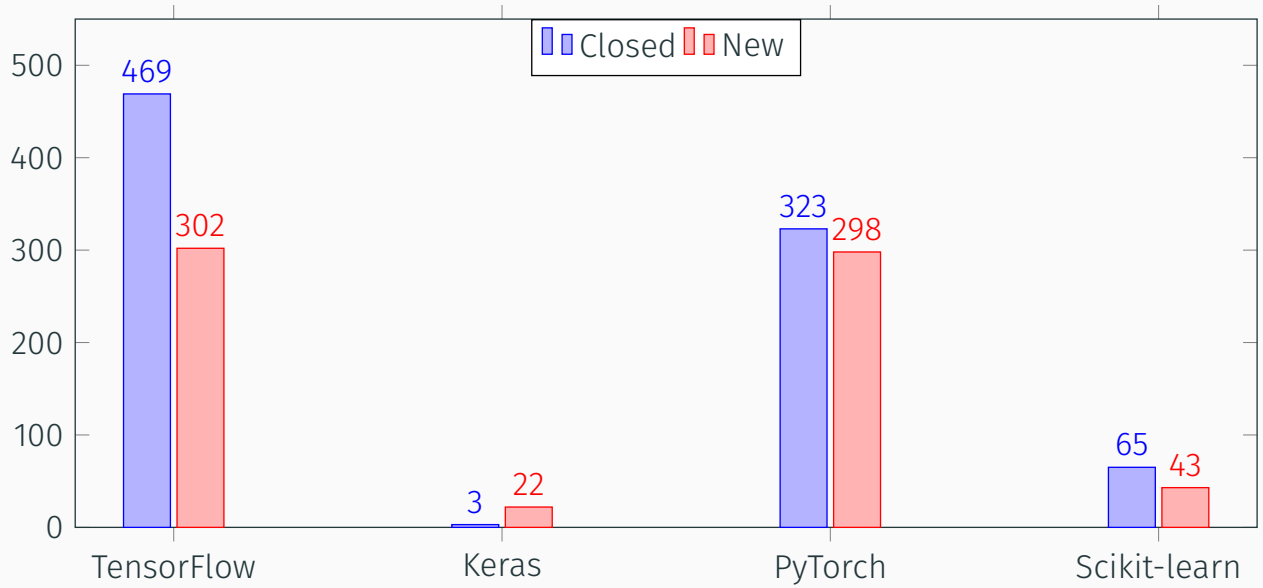

Figure 14: Monthly amount of new and closed issues for four packages(Nov, 2020 - Dec, 2020)

This comparison provides data that can reminds novices to use TensorFlow or PyTorch as soon as possible, because these two packages are activated currently.

Activity of Packages Cont.

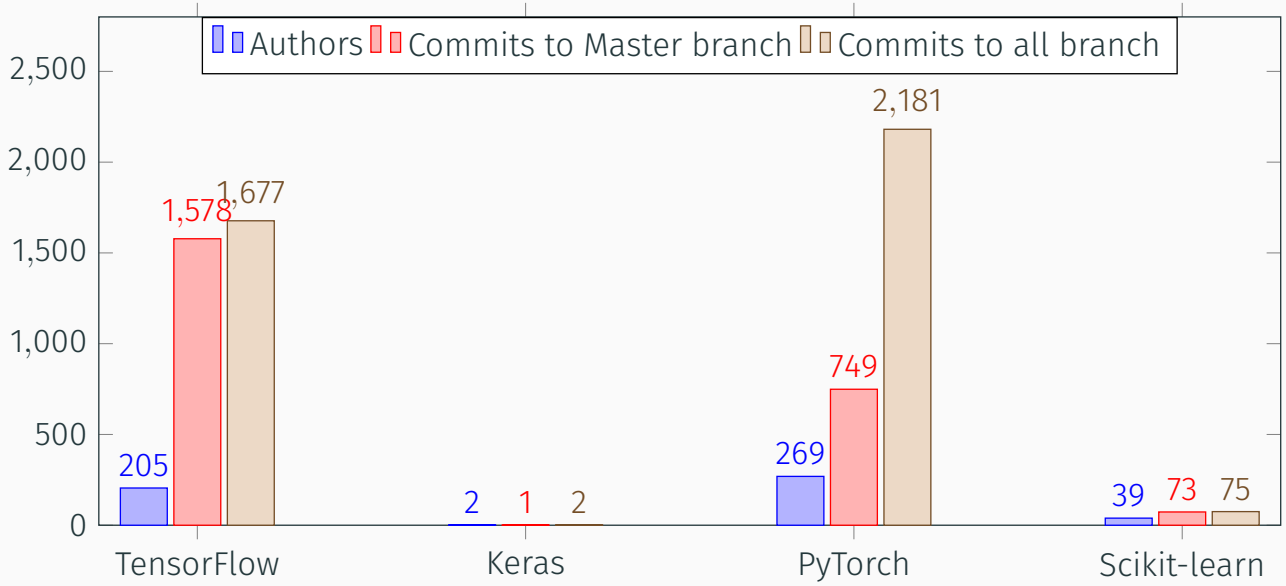

Figure 15: Modification Records for four packages(Nov, 2020 - Dec, 2020)

In summary, TensorFlow and PyTorch are more popular frameworks than others, and their friendly community ensure their users can easily get feedback from major developers or other users for the problem they meet.

Conclusions

Conclusions

Rubrics \ Frameworks	TensorFlow	PyTorch	Keras	Scikit-learn
Performance	90	**100**	78	0
Weight Initialization	**100**	87	**100**	0
Act./Loss Function Coverage	48	**100**	48	17
Opt. Method Coverage	**100**	**100**	64	27
Hardware/Platform Support	**100**	92	**100**	75
Language Support	**100**	60	40	20
Usability	31	30	81	55
Activity of Packages	**100**	**100**	25	50
Summary	**669**	**669**	536	244
Normalized Summary	**0.8363**	**0.8363**	0.67	0.305

Table 6: The grade table for four packages

We decide to calculate scores for each packages as a direct criterion to judge how packages' performance on different rubrics. And based on scores, analyze which package is the best one on doing machine learn.

Summary

For Scikit-learn, it is a very preliminary neural network library, and it cannot provide too many up-to-date features. However, due to its simpleness, it is very suitable for students who are beginner of neural network. For industrial level production, Scikit-learn is not a reasonable package for neural network.

For Keras, it can provide more cutting edge features with users. Compared with TensorFlow and PyTorch, it can construct the same neural network via less codes. This is more suitable for someone who want to exploit the advantages of neural network without firm coding skill. Due to the shortage of developers, it would become a problem about whether Keras would be continuously improving.

Summary Cont.

For PyTorch, due to the support of FAIR, it can provide the most cutting edge features and neural network models with users. The impressive computing capability for tensor also ensure the model training in a fast and efficient way. The active community provides user-friendly experience with users. For any project which put high priority on performance, the project recommends PyTorch.

For TensorFlow, the language and platform supports are more comprehensive than other packages. For any project required special language or platform, TensorFlow is the best choice. Since TensorFlow is maintained and developed by Google's teams, it would continuously provide top-to-art research findings with users, which means projects based on TensorFlow can easily switch to up-to-date neural network models.

Acknowledgements

The slides are written in LaTeX with theme: Metropolis and PolyU color scheme from Prof. Qu Xiaofeng, licensed under Creative Commons Attribution-ShareAlike 4.0 International License.

Contributions

Name	Student ID	Contributions
GUO Junpeng[1]	20097916G	Implement neural network models by PyTorch, write the report part 1, 2.2, 3.8, 4, organize and assign tasks to group members
ZHAO Shuyu	20068999G	Implement TensorFlow CNN, MLP, MLP Regression, write the representation slides, regulate the format of the report and the slides
YANG Zunrui	20073942G	Implement Keras CNN, MLP, MLP Regression, summarize language support, write 3.7 in report
HAO Peide	20103442G	PyTorch CNN&MLP,Code performs efficiency tests,part 3.1 and 3.2
CHEN Lu	20073318G	Implement Scikit-learn MLP and MLP Regression, write 2.4 and 3.7 part in report, write the representation slides
SUN Xu	20096765G	Collect the data information about TensorFlow and write 2.1 and 3.4 part in report
Zhao Jiayi	20074559G	Literature survey on Keras and write related code annotation, summarize hardware&platform compatibility, write 2.3&3.5 in report

Table 7: Group members' contributions

[1]Standing for group leader

References I


Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner.
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324, 1998.


Keiron O'Shea and Ryan Nash.
An introduction to convolutional neural networks.
arXiv preprint arXiv:1511.08458, 2015.


Leonardo Noriega.
Multilayer perceptron tutorial.
School of Computing. Staffordshire University, 2005.