Frame Works and Libraries of AI and Machine Learning: A Comprehensive Survey

Mr. Pawan Kumar[1], Prof. (Dr.) Durgesh Pant[2]

[1] *Department of Computer Science and Engineering, UOU, Haldwani, Uttarakhand, India*
Email: pawankmrarya@gmail.com
[2] *Department of Computer Science and Engineering, UOU, Haldwani, Uttarakhand, India*
Email: dpant@uou.ac.in

Abstract

Identifying visual patterns in an image is referred to as image classification, a key challenge in computer vision. This process analyzes the numerical characteristics of different image aspects and classifies the data into different categories [28]. In recent years, advanced techniques like artificial neural networks (ANN), fuzzy sets, and expert systems have been applied to image classification. However, these approaches often encounter limitations and tend to deliver lower accuracy.

A more advanced approach involves the use of deep convolutional neural network (CNN) architectures from deep learning, which have proven effective in addressing various machine learning challenges. In this study, image classification is conducted via a model similar to the LeNet network. The Keras library, a popular tool for deep learning, is employed in the process. This approach is evaluated for its accuracy and speed performance, yielding acceptable results.

Keywords: Deep learning (DL), Convolutional neural network (CNN), Artificial neural network (ANN), Keras, TensorFlow

1 Introduction

In the last ten years, deep learning has been successfully applied in several areas, including natural language processing (NLP), computer vision, voice recognition, and classification of images. Its success is largely due to its remarkable capacity to represent input data, achieved through the use of several layers of artificial neurons. Given the wide range of deep learning tools and hardware platforms available, selecting the most suitable tool for specific deep learning tasks can be challenging for end users.

In this paper, we have gathered as many existing frameworks and libraries as possible. Instead of detailing how to use each one, we focus on highlighting the unique feature of each framework and library. For example, we compare various deep learning frameworks on the basis of factors such as development, use case, ease of use (including deployment and performance), scalability

(such as top APIs such as Keras), and community support. This comparison helps us –establish guidance for researchers in this field.

2 Various frameworks and libraries

2.1 TensorFlow

TensorFlow was first released to the public in 2015, with its initial stable version debuting on February 11, 2017. Developed and maintained by Google, it has quickly emerged as one of the most popular frameworks for machine learning and deep learning initiatives. TensorFlow boasts an extensive library designed for large-scale machine learning and numerical computation.

TensorFlow is a flexible and scalable software library made for numerical calculations based on dataflow graphs. With the help of its related tools, neural networks and other machine learning models can be effectively created, trained, and implemented by users. [29] in real-world settings. NVIDIA's CUDA (Compute Unified Device Architecture) framework [30] and C++ are used to implement TensorFlow's basic algorithms. Although TensorFlow has APIs in several languages, the most comprehensive and reliable is the Python API. Officially supported languages also include JavaScript, C++, Go, and Swift. Additionally, TensorFlow integrates languages such as Java and R.

A graphical tool called TensorBoard was developed to facilitate the comprehension and debugging of TensorFlow scripts. Neural network models can be complex and difficult to manage, often leading to confusion. It allows users to easily visualize the computation graph of a model, track training metrics, and monitor parameter values.

TensorFlow is the most popular deep learning framework [1]. and has many characteristics that are unmatched; however, there are still numerous issues that need to be resolved, and the framework is still being developed. The benefits of TensorFlow are as follows:

- TensorFlow is a deep learning framework that can handle complex neural network topologies; it is efficient and flexible.
- Because of its capacity for large-scale data processing, it is well suited for distributed computing initiatives and large-scale dataset projects.
- Distributed computing support allows models to be trained over several GPUs and CPUs, which reduces calculation times and increases performance.
- It makes the process of creating complex AI systems easier with its wide ecosystem of prebuilt models, tools, and libraries.
- Large, vibrant community support for TensorFlow guarantees regular updates, bug fixes, and comprehensive documentation.
- TensorFlow, which is powered by Google, benefits from substantial support and continuous development from Google's AI experts.

- In addition to deep learning, TensorFlow provides numerous customization options for a broad variety of machine learning workloads.

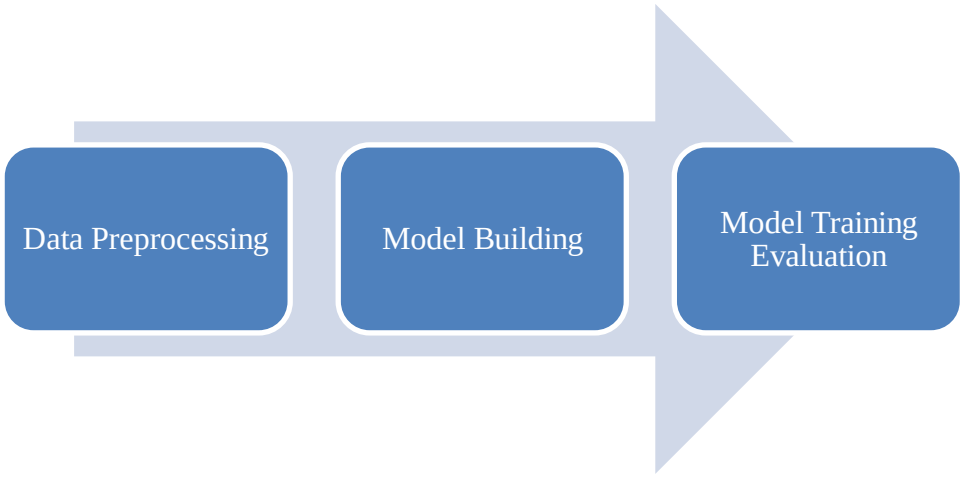

Fig. 1 - TensorFlow architecture consisting of three parts

The steps for installing TensorFlow via different methods are as follows:

1. Using pip (Python's package installer):

- Open your command prompt or terminal.
- The following command is used to install TensorFlow:

pip install TensorFlow

- For systems with GPU support, the following GPU version can be installed:

pip install TensorFlow-gpu

2. Using conda (Anaconda or Miniconda):

- The Anaconda Prompt or Terminal was opened.
- The following command is used to install TensorFlow:

  Conda install TensorFlow

- To install the GPU version, use:

conda install TensorFlow-gpu

3. From source:

- Ensure that you have Python and pip installed.
- Clone the TensorFlow repository from GitHub:

git clone https://github.com/TensorFlow/TensorFlow.git

- Navigate to the TensorFlow directory and run the following commands:

cd TensorFlow

pip install -r requirements.txt

python setup.py install

- This method is recommended for advanced users who need to customize TensorFlow.

4. Using Docker:

- Docker allows you to containerize your TensorFlow installation.
- Pull the TensorFlow Docker image by running:

docker pull TensorFlow/TensorFlow

- For the GPU version:

docker pull TensorFlow/TensorFlow:latest-gpu

- You can then run TensorFlow in a Docker container.

5. Using a prebuilt distribution:

- TensorFlow can be installed as part of a Python distribution such as Anaconda, which may include TensorFlow or allow easy installation via the Anaconda Navigator interface.

These methods offer flexibility depending on the development environment, whether Python, Anaconda, Docker, or TensorFlow is used from the source.

2.2 Keras

TensorFlow and other lower-level frameworks can be used with Keras, a high-level neural network API developed in Python. Designed for ease of use, modularity, and extensibility, Keras aims to facilitate rapid experimentation and quick prototyping. It supports various neural network components, such as dense layers, convolutional layers, recurrent layers, dropout layers, and their derivatives. The library dynamically manages resources like the GPU (graphics processing unit) and CPU (central processing unit) to optimize performance.

Keras makes it easier to swiftly create and train models. With Keras, testing models is simple and simply requires metrics for assessment, the number of training epochs, and specifications. Because of its simplicity, most deep learning models can be implemented with a lot less lines of code. By utilizing Keras, users can enhance productivity and allocate more time to designing improved deep learning algorithms and other important tasks. The sequential model API allows users to generate standard models with a few lines of code.

Moreover, Keras can also create complex graph-based structures, allow for layer reuse, and develop models that function like Python functions using its Functional API. It is flexible enough to integrate new or experimental deep learning frameworks and layers based on its concept and layer classes. Keras has emerged as a dominant tool for build and developing neural network models with minimal code. Its customizable Model and Layer classes, along with its API, enable users to create increasingly intricate and sophisticated models.

TensorFlow, Keras, PyTorch and Scikit-Learn are the most extensively used frameworks for deep learning. Learning a new framework can be challenging, so many people struggle with deciding between them. Initially, these frameworks differed significantly in terms of design, paradigm, and syntax. However, they have evolved considerably over time, adopting many of each other's strengths, making them more similar than before. Numerous internet comparisons between the two systems are out of date and do not fairly represent the situation as it stands right now. Figure 2. Show the recent Google search trends between these frameworks.

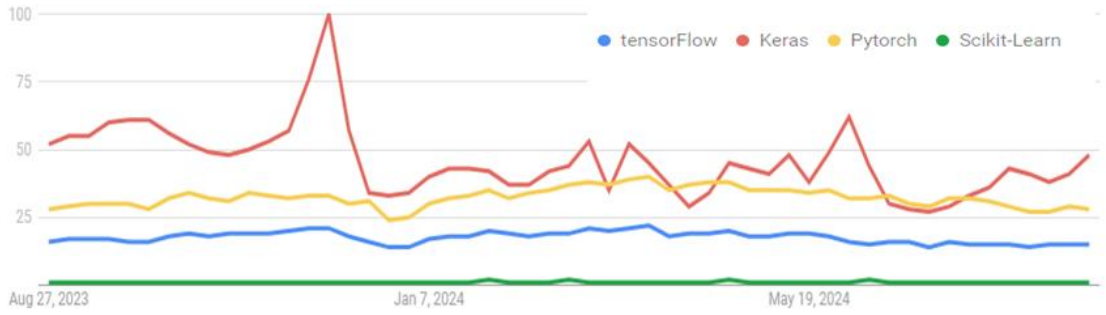

Fig. 2 – Source: Google search trends

There are several justifications for becoming proficient with Keras. There are a few causes for this.

- It is adaptable and follows the growing complexity disclosure principle: simple, rapid tasks may be completed, and large workflows can be accomplished by taking small, unambiguous stages.

- It is intended to be easy to use and straightforward. It is a great substitute for people who are new to deep learning because it captures the majority of TensorFlow's low-level complexity.
- It enables speedy neural network prototyping so that different topologies can be tested correctly.
- It has been integrated as the official high-level API in TensorFlow from version 2.0, guaranteeing compatibility and synergy between the two.
- Its code is frequently more concise and understandable than TensorFlow code.
- Using the scalability and performance that lead the industry, it is utilized by companies such as Waymo, YouTube, and NASA.

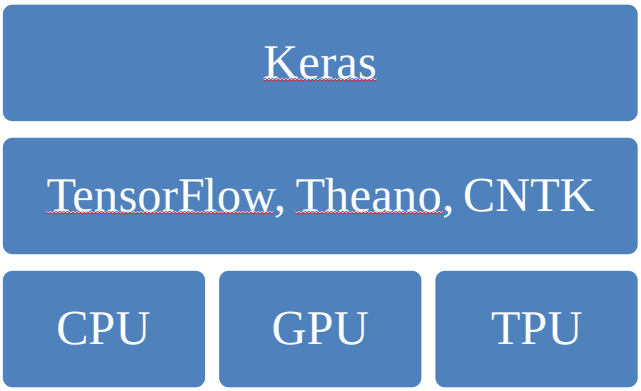

Fig. 3 - Keras lies on top of other deep learning models [1]

The steps for installing Keras via different methods are as follows:

1. Using pip (Python's package installer):

- Open your command prompt or terminal.
- The following command is used to install Keras:

pip install keras

- Keras is now part of TensorFlow, so installing TensorFlow will also give you access to Keras:

pip install TensorFlow

2. Using conda (Anaconda or Miniconda):

- The Anaconda Prompt or Terminal was opened.

- The following command is used to install Keras:

Conda install keras

- Alternatively, if you want to install TensorFlow (which includes Keras):

Conda install TensorFlow

3. From source:

- Ensure that you have Python and pip installed on your system.
- Clone the Keras repository from GitHub:

git clone https://github.com/keras-team/keras.git

- Navigate to the Keras directory and run the following commands:

cd keras

pip install -r requirements.txt

python setup.py install

- This method is ideal for developers who need to modify Keras.

4. Using a prebuilt distribution:

- If Anaconda is used, Keras can be installed directly from the Anaconda Navigator interface by searching for Keras or TensorFlow in the package manager and installing it with a few clicks.

Each of these methods provides a straightforward way to install Keras, depending on the specific development environment and requirements.

2.3 PyTorch

PyTorch is an open-source machine learning framework developed by Facebook's AI research division. Its flexible dynamic computation graph makes it easy to utilize for debugging and model development.

PyTorch is a Python package that offers two key advanced capabilities. The first is GPU-based tensor calculation [1], which is similar to NumPy and may generally be used in its place. Constructing a novel dynamic neural network is the second goal. As previously noted, Torch is great, but since it is built in Lua, it has a drawback: Lua has thread restrictions. While Python has an advantage in this regard, multithreading does not greatly benefit from this. Python's simplicity is made available to PyTorch users through the integration of Torch.

Most frameworks, such as TensorFlow, Theano, Caffe, and CNTK, are static, requiring users to build a neural network and reuse the same structure [1], meaning that any changes to the network require starting from scratch. On the other hand, PyTorch employs a method known as reverse-mode auto differentiation that enables users to alter network performance without any overhead or delay. Although this method is not unique to PyTorch, it is among the quickest implementations on the market, which gives it a major edge over other frameworks.

Key features of PyTorch:

- **Dynamic computation graph:** Often referred to as "define-by-run," this feature makes it simple to modify the graph while it is being created.
- **Tensors and Autograd:** This package allows gradient computation for n-dimensional arrays (tensors) via automatic differentiation (AutoGrad).
- **Extensive Library:** This contains many preconstructed layers, optimizers, and loss functions.
- **Interoperability:** NumPy, SciPy, and other Python libraries can be readily integrated with it.
- **Community Support**: A robust network of tools and extensions for the community.

The steps for installing PyTorch via various methods are as follows:

1. Using pip (Python's package installer):

- Open your command prompt or terminal.
- Visit the official PyTorch website to find the specific installation command for your environment (OS, Python version, and whether you want GPU support).
- Typically, the command looks like this for the CPU version:

pip install torch torchvision torchaudio

- For the GPU version with CUDA support, use:

pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118

- Replace cu118 with the appropriate CUDA version if necessary.

2. Using conda (Anaconda or Miniconda):

- The Anaconda Prompt or Terminal was opened.
- Visit the official PyTorch website to select the right command on the basis of your setup.

- A typical command might be as follows:

conda install pytorch torchvision torchaudio cpuonly -c pytorch

- For GPU support, including CUDA:

conda install pytorch torchvision torchaudio cudatoolkit=11.8 -c pytorch

- The cudatoolkit=11.8 is replaced with the appropriate CUDA version if necessary.

3. Using Docker:

- Docker allows you to containerize your PyTorch installation.
- Pull the PyTorch Docker image by running:

docker pull pytorch/pytorch

- For a specific version or with CUDA support, refer to the PyTorch Docker Hub for the correct tag:

docker pull pytorch/pytorch:latest-cuda11.8-cudnn8-devel

- Then, PyTorch is run in a Docker container.

4. From source:

- Ensure that you have Python, pip, and other build tools installed.
- Clone the PyTorch repository from GitHub:

git clone --recursive https://github.com/pytorch/pytorch

- Navigate the PyTorch directory and build it:

cd PyTorch

python setup.py install

- This method is best for developers who need to modify PyTorch or want to build it with specific configurations.

5. Using a prebuilt distribution:

- PyTorch can also be installed via prebuilt Python distributions such as Anaconda, where it can be found in the Anaconda Navigator's package manager and installed with a few clicks.

Each method allows you to install PyTorch in a way that best suits your environment and needs. For most users, using pip or condas with the appropriate settings from the PyTorch website is the easiest approach.

2.4 Scikit-learn

A well-liked open-source machine learning library for Python is called Scikit-learn, or sklearn. It offers a large selection of algorithms, which facilitates model training and evaluation. NumPy, Matplotlib, and SciPy are among the fundamental Python libraries upon which Scikit-learn is based. These tools are perfectly integrated with Scikit-learn, creating a comprehensive machine learning environment.

Here, are the use cases of scikit-learn:

- Spam detection,
- Sentiment analysis
- Image classification.
- Housing prices
- Stock market trends analysis.
- Customer segmentation,
- Image segmentation
- Recommendation systems.
- The dataset dimensionality is reduced,
- Facilitating easier visualization and analysis of complex data.
- Text classification
- Sentiment analysis
- Fraud detection
- Network security
- Quality control.
- Enhanced predictive performance.

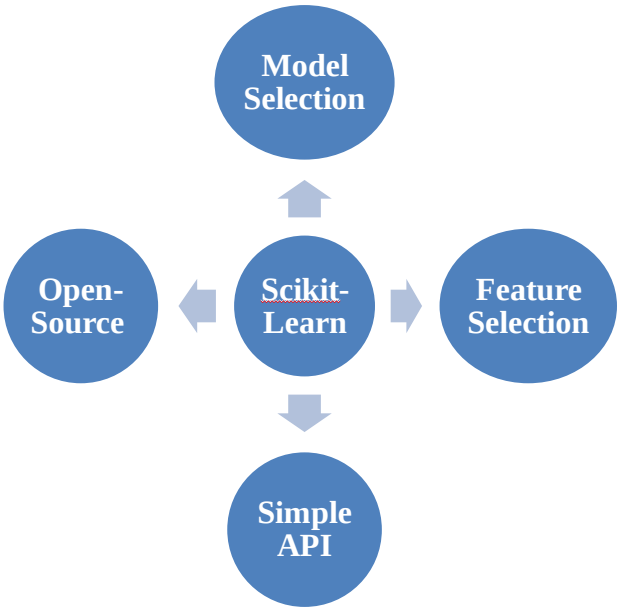

Fig. 4 - Features of scikit-learn

The steps to install scikit-learn via different methods are as follows:

1. Using pip (Python's package installer):

- Open your command prompt or terminal.
- Execute the following command:

pip install scikit-learn

2. Using conda (Anaconda or Miniconda):

- If Anaconda or Miniconda is used, the Anaconda Prompt or Terminal is opened.
- Execute the following command:

conda install scikit-learn

3. From source:

- Make sure Python and pip are installed on your system
- Download the source code from the official website scikit-learn GitHub repository.
- Navigate to the directory containing the source code.
- Run the following commands:

pip install -r requirements.txt

python setup.py install

4. Using a prebuilt distribution:

- Scikit-learn can also be installed as part of a larger Python distribution such as Anaconda, which comes with scikit-learn preinstalled.
- Anaconda or Miniconda is downloaded and installed, and scikit-learn is available by default.

These methods provide flexibility depending on the development environment and preferences.

Table 1 – Comparison between Deep Learning Frameworks

Aspect	**TensorFlow**	**Keras**	**PyTorch**	**Scikit-Learn**
Developed By	Google Brain	François Chollet (now part of TensorFlow)	Facebook	David Cournapeau
User-Friendly	Complex	Easy to use and straightforward to implement.	Easy to understand and use.	user-friendly interface
Flexibility	Highly adaptable	Less adaptable, mainly suited for standard layers and models.	Highly adaptable	adaptable
Deployment	Wide support (TensorFlow Lite, TensorFlow Serving, TensorFlow JS)	Uses TensorFlow for deployment	Uses TorchServe	Modular, and Flexible Toolkit
Performance	Performance-enhancing and supporting dispersed training	optimized using the backend of TensorFlow	Research-focused	Industry and research focused.
Visualization	TensorBoard	Supports TensorBoard	Integrates with TensorBoard	defines a simple API

API level	High level and low level	High-level	Low level	High-level
Use Case	Ideal for complex, large-scale projects	Suitable for rapid prototyping and experimentation	Ideal for complex projects	Suitable for experimentation
Data Handling	Using the tf.data API, Advanced data handling	Using built-in methods, Simple data handling	Efficient data handling	Smooth data handling
Community Support	Large community , industry-adopted	Large community, integrated within TensorFlow	Growing, academia-focused	The community support is active, but it is not as widespread as TensorFlow's.

Conclusion

The rapid growth of deep learning frameworks has been fueled by the growing prominence of deep learning in computer vision and natural language processing (NLP) in recent years. However, there is still a lack of research on how to choose the most appropriate framework [1] or recognize emerging trends based on the development of these frameworks. This survey addresses this gap by evaluating widely used frameworks, presenting extensive performance comparison data, and offering various evaluation criteria to guide researchers in choosing the appropriate framework.

References

1. Wang, Z., Liu, K., Li, J., Zhu, Y., & Zhang, Y. (2019). Various Frameworks and Libraries of Machine Learning and Deep Learning: A Survey. Archives of Computational Methods in Engineering, 31(1), 1–24. https://doi.org/10.1007/s11831-018-09312-w
2. Pang, B., Nijkamp, E., & Wu, Y. N. (2019). Deep Learning With TensorFlow: A Review. In Journal of Educational and Behavioral Statistics: Vol. XX–X (pp. 1–22). https://doi.org/10.3102/1076998619872761
3. Shi, S., Wang, Q., Xu, P., & Chu, X. (2016). Benchmarking State-of-the-Art Deep Learning Software Tools. https://doi.org/10.1109/ccbd.2016.029

4. Chicho, B. T., & Sallow, A. B. (2021). A Comprehensive Survey of Deep Learning Models Based on Keras Framework. Journal of Soft Computing and Data Mining, 2(2). https://doi.org/10.30880/jscdm.2021.02.02.005
5. Gao, X., Ramezanghorbani, F., Isayev, O., Smith, J. S., & Roitberg, A. E. (2020). TorchANI: A Free and Open Source PyTorch-Based Deep Learning Implementation of the ANI Neural Network Potentials. Journal of Chemical Information and Modeling, 60(7), 3408–3415. https://doi.org/10.1021/acs.jcim.0c00451
6. Ertam, F., & Aydin, G. (2017). Data classification with deep learning using TensorFlow. 2017 International Conference on Computer Science and Engineering (UBMK). https://doi.org/10.1109/ubmk.2017.8093521
7. Nagisetty, A., & Gupta, G. P. (2019). Framework for Detection of Malicious Activities in IoT Networks using Keras Deep Learning Library. https://doi.org/10.1109/iccmc.2019.8819688
8. Atienza, R. (2020). Advanced Deep Learning with TensorFlow 2 and Keras. In Packt publishing eBooks. http://ofppt.scholarvox.com/catalog/book/88882283
9. Wan, H. (2019). Deep Learning:Neural Network, Optimizing Method and Libraries Review. https://doi.org/10.1109/icris.2019.00128
10. Jain, A., Awan, A. A., Subramoni, H., & Panda, D. K. (2019). Scaling TensorFlow, PyTorch, and MXNet using MVAPICH2 for High-Performance Deep Learning on Frontera. https://doi.org/10.1109/dls49591.2019.00015
11. Parvat, A., Chavan, J., Kadam, S., Dev, S., & Pathak, V. (2017). A survey of deep-learning frameworks. 2017 International Conference on Inventive Systems and Control (ICISC). https://doi.org/10.1109/icisc.2017.8068684
12. Zeroual, A., Derdour, M., Amroune, M., & Bentahar, A. (2019). Using a Fine-Tuning Method for a Deep Authentication in Mobile Cloud Computing Based on TensorFlow Lite Framework. https://doi.org/10.1109/icnas.2019.8807440
13. Alahmari, S. S., Goldgof, D. B., Mouton, P. R., & Hall, L. O. (2020). Challenges for the Repeatability of Deep Learning Models. IEEE Access, 8, 211860–211868. https://doi.org/10.1109/access.2020.3039833

14. Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient Processing of Deep Neural Networks: A Tutorial and Survey. Proceedings of the IEEE, 105(12), 2295–2329. https://doi.org/10.1109/jproc.2017.2761740
15. Stancin, I., & Jovic, A. (2019). An overview and comparison of free Python libraries for data mining and big data analysis. https://doi.org/10.23919/mipro.2019.8757088
16. Jiang, Z., & Shen, G. (2019). Prediction of House Price Based on The Back Propagation Neural Network in The Keras Deep Learning Framework. https://doi.org/10.1109/icsai48974.2019.9010071
17. Singhla, R., Singh, P., Madaan, R., & Panda, S. (2021). Image Classification Using Tensor Flow. https://doi.org/10.1109/icais50930.2021.9395939
18. Kim, S., Wimmer, H., & Kim, J. (2022). Analysis of Deep Learning Libraries: Keras, PyTorch, and MXnet. https://doi.org/10.1109/sera54885.2022.9806734
19. Shushkevich, E., Alexandrov, M., & Cardiff, J. (2021). Detecting fake news about Covid-19 using classifiers from Scikit-learn. 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT). https://doi.org/10.1109/csit52700.2021.9648767
20. Bhardwaj, P., Choudhury, C., & Batra, P. (2023). Automating Data Analysis with Python: A Comparative Study of Popular Libraries and their Application. https://doi.org/10.1109/ictacs59847.2023.10390032
21. A Huang, X., Du, X., Liu, H., & Zang, W. (2021). A Research on Face Recognition Open Source Development Framework Based on PyTorch. https://doi.org/10.1109/isctis51085.2021.00077
22. Ji, J., Kong, W., Tian, J., Gu, T., Nie, Y., & Kuang, X. (2023). Survey on Fuzzing Techniques in Deep Learning Libraries. https://doi.org/10.1109/dsc59305.2023.00073
23. Lyzhin, D., Volodchenko, I., & Petrova, L. (2020). Application for Recognition of Transport Using Pytorch Library. 2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon). https://doi.org/10.1109/fareastcon50210.2020.9271215
24. Lambeta, M., Xu, H., Xu, J., Chou, P. W., Wang, S., Darrell, T., & Calandra, R. (2021). PyTouch: A Machine Learning Library for Touch Processing. https://doi.org/10.1109/icra48506.2021.9561084

25. Riba, E., Mishkin, D., Ponsa, D., Rublee, E., & Bradski, G. (2020). Kornia: an Open Source Differentiable Computer Vision Library for PyTorch. https://doi.org/10.1109/wacv45572.2020.9093363
26. Chirodea, M. C., Novac, O. C., Novac, C. M., Bizon, N., Oproescu, M., & Gordan, C. E. (2021). Comparison of TensorFlow and PyTorch in Convolutional Neural Network - based Applications. https://doi.org/10.1109/ecai52376.2021.9515098
27. Cheng, J., Wang, P. S., Li, G., Hu, Q. H., & Lu, H. Q. (2018). Recent advances in efficient computation of deep convolutional neural networks. Frontiers of Information Technology & Electronic Engineering, 19(1), 64–77. https://doi.org/10.1631/fitee.1700789
28. Sharma, R., & Singh, A. (2021). Image Pre-Processing and Paddy Pests Detection Using Tensorflow. In Advances in medical technologies and clinical practice book series (pp. 131–139). https://doi.org/10.4018/978-1-7998-7188-0.ch010
29. Omar, H. K., Frikha, M., & Jumaa, A. K. (2024). PyTorch and TensorFlow Performance Evaluation in Big Data Recommendation System. Ingénierie Des Systèmes D Information, 29(4), 1357–1364. https://doi.org/10.18280/isi.290411
30. Perera, D., Reisenhofer, E., Hussein, S., Higgins, E., Huber, C. D., & Long, Q. (2023). CATE: A fast and scalable CUDA implementation to conduct highly parallelized evolutionary tests on large scale genomic data. Methods in Ecology and Evolution, 14(8), 2095–2109. https://doi.org/10.1111/2041-210x.14168