ANALYSIS OF THE MOST POPULAR LIBRARIES AND PLATFORMS THAT CAN BE USED TO CREATE A MODEL FOR THE INTELLIGENT RECOGNITION OF MICROFOSSILS

Cristian CUDALBU[1], Eliza ANTON[1], Constantin LAZAR[1]

[1] National Institute of Marine Geology and Geo-Ecology (GeoEcoMar), 23-25 Dimitrie Onciul St., 024053 Bucharest, Romania
e-mail: c.cudalbu@geoecomar.ro, antoneliza@geoecomar.ro, lazar.constantin@geoecomar.ro

DOI: 10.xxxx

Abstract. This paper presents an analysis of the most common software tools (libraries and platforms) used in the field on AI image processing by building deep learning models. The most useful characteristics for each tool are analyzed for providing a suitable candidate for a deep learning model in the field of intelligent image recognition for microfossils.

Key words: machine learning, image recognition, microfossils

1. INTRODUCTION

Microfossils are a heterogeneous group of fossils, studied in micropaleontology. These, unlike other types of fossils, are not grouped according to phylogenetic relationships but based on their generally small size (micron to centimeter). Microfossils are the most important group among all fossils, being extremely useful for dating according to age, correlation and paleoenvironmental reconstruction, very important in the oil, mining, engineering and environmental industries, as well as in general geology. Because they usually occur in large numbers in sediments, they are the most abundant and most easily accessible fossils. For micropaleontological investigations, rock samples must be processed for separating microfossils; afterwards, they are microscopically analyzed for generic and specific determination.

The calcareous nanoplankton is a major component of oceanic phytoplankton. The evolutionary model of this group of marine organisms and its current distribution throughout the marine world are extremely useful in various fields of research, such as: marine geology, biogeochemistry and paleontology (Brasier, 1980; Haq & Boersma, 1978).

Since its discovery in the mid-nineteenth century (Ehrenberg, 1836), the calcareous nanoplankton has been important for paleontological studies (to support the evolutionary hypothesis) and biostratigraphy studies (for accurate relative dating of marine sedimentary successions), being intensively used for paleoecology and paleogeographic reconstructions. Because calcareous nanoplankton is a group of organisms extant, one of the most diverse and widespread in the world of marine phytoplankton, it is used for marine biology and geological investigations as well as in analyzes and predictions of the evolution of the marine environment.

The average size of calcareous nanoplankton is between 5-15 microns. A more restrictive definition (Tappan, 1980) considers that taxa smaller than 2 µm can be classified as ultramicroplankton, and those between 2-20 µm belong to calcareous nanoplankton. Fossils of calcareous nanoplankton are also described as calcareous nannofossils (or simply nannofossils). The term nannoflora was also used for describing this group of unicellular algae (Thierstein & Young, 2004; Melinte, 2004).

The identification and classification of microfossils is the main direction of study in the field of micropaleontology. Conventionally, paleontologists qualitatively identify and classify microfossils based on morphological features, by studying micron-sized organisms, *i.e.*, calcareous nannofossils, diatoms and radiolarians, or microfossils, such as foraminifers and ostracods, under an optical or electron microscope. These traditional methods are time consuming and require considerable expertise due to the large number and high diversity of microfossils. In some cases, paleontologists apply methods of geometric morphometry to identify and classify fossils (Saraswati & Srinivasan, 2016; Cui *et al.*, 2019). However, these methods are not fully automated, and a large number of indexes and benchmarks must be defined for measurement.

In recent years, machine learning and especially deep learning have led to excellent results in the classification of microfossils. A deep learning method can implement the „end-to-end" classification procedure, which is more objective and not limited to one type of fossil (Brocklehurst *et al.*, 2018).

Machine learning models and algorithms can help us quickly and easily implement specific image processing functionalities. However, building a personalized machine learning model or a basic neural network requires a lot of resources and a high level of technological expertise. With the help of open-source tools, libraries and frameworks that exist and are presented herein, one can choose to implement in their own software model the best solution for the intelligent automatic recognition of microfossil images.

2. ANALYSIS

2.1. TensorFlow

TensorFlow is an open-source platform for machine learning. It has a comprehensive and flexible ecosystem of tools, libraries and community resources, which allows researchers to use the latest technologies in ML (Machine Learing) and developers to easily build and implement ML-based applications (https://www.tensorflow.org).

The platform was developed by Google.com in 2015, initially for internal use, which was later launched as an open-source software library gaining much popularity to date.

The name "**TensorFlow**" comes from the two specific programming features:

- **Tensors**, which are multidimensional data arrays;
- **Data flow graphs**, are a kind of neural network that takes into consideration the previous states of the system.

TensorFlow is designed on the operations that the data flow graphs perform on such data arrays. A compiler XLA, which is specific to these graphs, has been added as part of the updates to the last versions of TensorFlow, and it allows for better performance (Sydorenko, 2021).

Thanks to its extensive library, TensorFlow can be used for efficient design of models that can be trained on standard datasets. This can significantly save time and resources for certain AI projects.

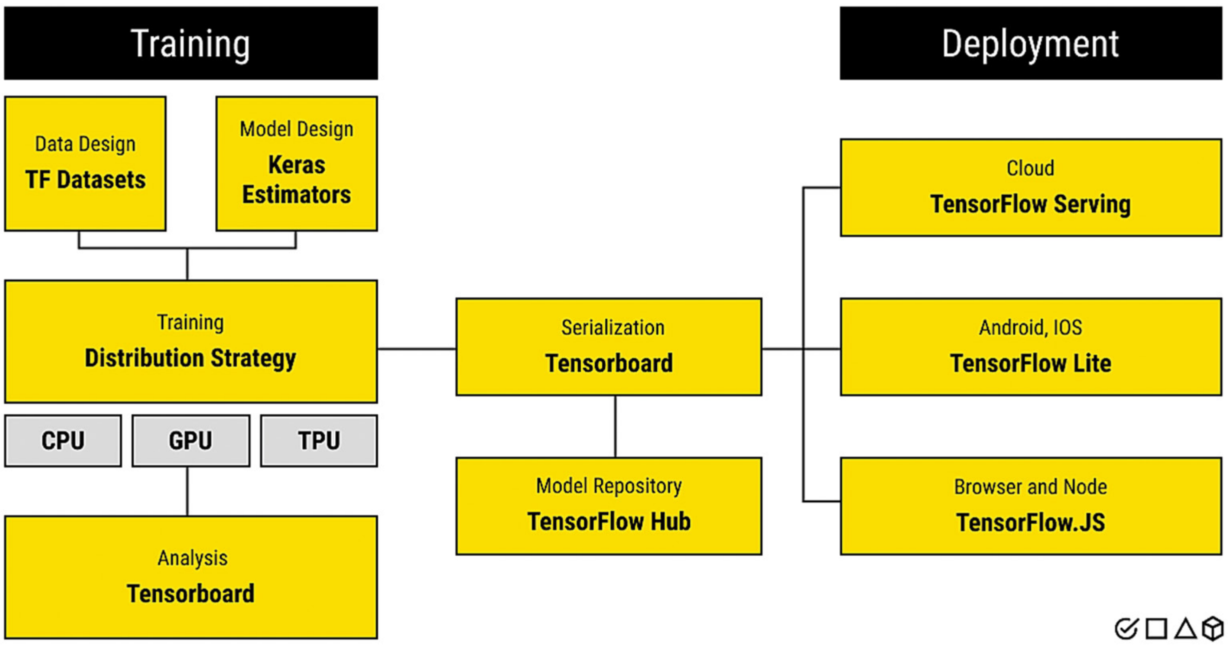

Fig. 1. Training and deploying an ML model with TensorFlow (after Sydorenko, 2021).

In addition, given the open-source basis of the tool, developed modules are permanently added to the core library, enhancing it and thus facilitating the work of ML platform developers.

Basically, TensorFlow has a set of features that favorably distinguish this software from many of its competitors. First, it has auto-differentiation, which was already accessible in the first version of the platform. Then there is a tool for viewing the model in TensorBoard, which makes this software significantly less complicated than some of its direct competitors. It also provides some of the best support services in the software industry for tools using Python as programming language (Sydorenko, 2021).

2.1.1. TensorFlow: Main Features

TensorFlow uses Python, for easy communication between software and users but was written in another programming language, namely C++ (Fig. 1). This language is one of the most powerful existing due to the direct communication through the instruction sets with the CPU unit (no need for a support framework like .Net or JVM), which is extremely advantageous when we talk about the need to train a complex machine learning algorithm using TensorFlow.

Due to the fact that TensorFlow allows the use of GPU instead of processors to train our ML algorithm, it increases the speed of the learning process. This is done by combining C ++ with Cuda technology (from Nvidia). Along with this comes a first issue in using it: although we can use almost any platform from Windows, macOS or Ubuntu Linux to iOS and Android in order to build a model using TensorFlow, Cuda has certain requirements for package and kernel versions if we want to get the best performance.

2.1.2. TensorFlow: How does it work?

TensorFlow is a very easy to use platform for designing, learning and implementing the ML algorithm with its main characteristics being (Fig. 1):

- TensorFlow Serving: a complete data loading system for ML algorithms designed for production environments.
- TensorFlow Lite: a simplified version of the platform that allows you to work with it on mobile devices.
- Fast execution: programming environment with an intuitive interface that does not build graphics, but evaluates operations immediately.
- Operations estimation: Prefabricated APIs used for training, evaluation, prediction and data export.
- Pre-compiled datasets: ready-to-use datasets that can be used with TensorFlow, but also with other machine learning frameworks (eg Jax).

The list of TensorFlow features listed here is by no means complete, but it gives us a general idea of how this platform works.

2.1.3. Advantages TensorFlow

- Open source. TensorFlow is free, can be downloaded and used without the need to purchase a software license or any additional costs (maintenance, support). This is probably what makes it so popular combined with the size of the ecosystem of TensorFlow tools, software libraries and resources.
- Platform flexibility. TensorFlow can be used on any hardware platform (Fig. 2), be it a processor (CPU), a GPU or a TPU using the advantageous features of each. In addition, TensorFlow allows you to work in Google Cloud, which provides an added benefit of speed and access to all parties involved in a single ML project (Sydorenko, 2021).

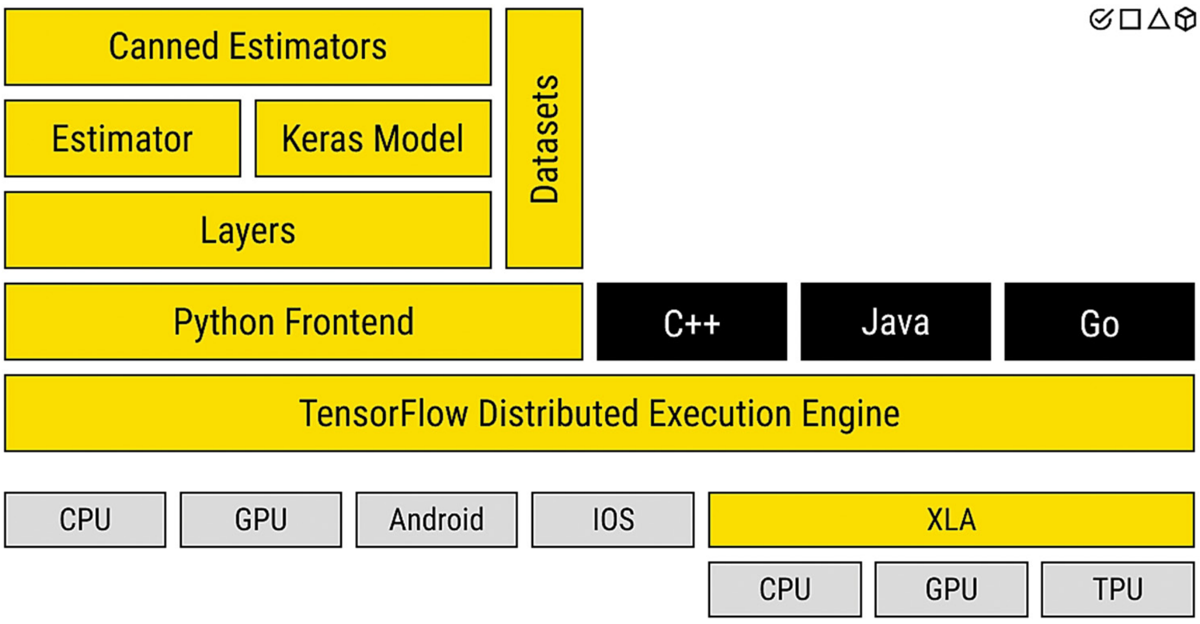

Fig. 2. Characteristics of TensorFlow (after Sydorenko, 2021).

- Scalability. It easily allows the construction of an incipient (test) model and then its expansion as the needs and complexity of operations increase.
- Allows experimentation. TensorFlow is also flexible in terms of designing the model architecture. This is a benefit for ML applications involving intelligent image recognition
- Feedback-based updates. Unlike many other open-source codes, TensorFlow is updated based on user feedback. For example, TensorFlow 2.0 was created to cover the most popular user requests for adjustments and changes. Future updates are also possible, making the TensorFlow platform an important tool for ongoing AI projects.
- Moderation in the use of computational resources (computational). TensorFlow's C ++ base allows more flexible use of hardware resources for a variety of tasks in machine learning. This speed up the training process without the need for excessive computing power.

2.1.4. Disadvantages TensorFlow

- Functionality of software updates. The new version of TensorFlow 2.0, although built on user feedback, contains some challenges in using the new implemented features. This makes compatibility between the first version of TensorFlow and TensorFlow 2.0 problematic. In most cases the algorithms written in the first version may require a full change in the newer version of TensorFlow.
- Reliability issues. While many may be tempted to continue working with the original version of TensorFlow, it may be less secure and reliable. There have been quite a few cases of memory leaks that have prevented and significantly affected the development process.
- Syntax complexity. While TensorFlow is built to communicate with users on Python, its syntax is somewhat different from classic Python and can be confusing. There is the possibility to overcome the complexities by using TensorFlow superstructures such as Keras which allows a simpler approach to building ML models.
- Operational instability. Despite its advantages of speed and flexibility, TensorFlow is still prone to crash, especially for more complex architectures. This can lead to loss of work and valuable time spent restarting work sessions.

2.2. OpenCV

OpenCV (Open-Source Computer Vision Library) is an open-source software library built to provide a common infrastructure for ML applications comprising over 2500 ML-optimized algorithms, in particular a comprehensive set of both classic and new generation machine learning algorithms. These algorithms can be used to detect and recognize human faces, identify objects, classify human actions into videos, track camera movements, track moving objects, extract 3D models of objects, and more. OpenCV has a community of over forty - seven thousand users and the estimated number of downloads exceeds eighteen million. The library is widely used in companies, research groups and by government agencies (www.opencv.org/about/).

2.2.1. OpenCV Main Features

OpenCV specializes mainly in image processing, being one of the most used libraries in ML and computer vision models. Users can perform several processing operations, such as histograms, color space conversion, geometric image transformations, image filtering, and so on. They can also perform resizing and other image processing techniques on their images using simple functions.

The main modules of Open CV (which also describe the main functionalities) are:

- Basic functions - this module covers basic data structures such as scalar, point, range, etc., which are used to build OpenCV applications. In addition, it includes the multidimensional Matrix Array, which is used to store images. In the Java library of OpenCV, this module is included as a package named org.opencv.core.
- Image processing - this module covers various image processing operations, such as image filtering, geometric image transformations, color space conversion, histograms, etc. In the Java library of OpenCV, this module is included as a package named org.opencv.imgproc.
- characteristics2d - This module includes the concepts of detection and description of two-dimensional characteristics. In the Java library of OpenCV, this module is included as a package named org.opencv.features2d.
- Objdetect - This module includes detecting objects and instances in predefined classes, such as trees, people, cars, animals, etc. In the Java library of OpenCV, this module is included as a package named org.opencv.objdetect.

2.2.2. OpenCV How does it works?

OpenCV is a library mainly focused on computer vision applications. This is an interdisciplinary field that deals with how computers can learn to understand the visualizations of the surroundings as close as possible to reality. After achieving this conceptual perspective, we move on to automating tasks or performing the desired action by the user.

OpenCV API allows setting up a line of learning for ML projects in three easy steps:

1. I / O. Upload data from image files, videos, capture devices.
2. Performing feature extraction. OpenCV contains a long list of algorithms, so there is no need to implement additional ones.
3. Application of machine learning algorithms for decision making, recognition and detection of objects.

2.2.3. Advantages OpenCV

- A large number of algorithms included: OpenCV offers access to over 2,500 classic and state-of-the-art algorithms. Using this library, users can perform various tasks, such as removing the red-eye effect from images, extracting 3D models of objects, tracking eye movements, and so on.

- Extensive use: Large companies such as IBM, Google, Toyota or startups such as Zeitera and Applied Minds use OpenCV for multiple tasks. In this way, users are assured that they have access to a library that is used by the most prestigious government institutions and enterprises in their fields of activity. In the OpenCV community, users can ask for help and help other developers. This gives developers easy access to library information and code snippets developed by other members of the community.
- Efficient solution: OpenCV provides algorithmic efficiency mainly for real-time instruction processing. Moreover, it has been designed in a way that allows it to take advantage of GPU hardware acceleration and multi-core CPU systems.

2.2.4. Disadvantages OpenCV

- Reduced learning capabilities (training) - OpenCV is a library of pre-programmed algorithms that can only be applied to a deep learning model that has already gone through the training stage in detecting objects of interest.
- Operational instability – crashes and freezes can occur quite often depending on the complexity of the problem addressed and the hardware capabilities.
- Poor integration with other applications - due to the fact that OpenCV has its own extensive library, conflicts may arise in the operation with other external applications if their integration is desired.

2.3. Keras

Keras is an open-source Python tool library for building deep learning models. It is an excellent option because it streamlines the construction of a deep learning model from scratch. Keras is simple to use and ideal for fast modeling of many types of neural networks.

TensorFlow was used to build the library, which is now fully embedded in it. This implies that we can design the Keras deep learning model, which has a much easier to use interface, and then integrate a specific TensorFlow functionality or feature into this model (Fawad Malik, 2021).

2.3.1. Keras Main Features

The main features of Keras relevant for this article are:

- Modularity: Keras is modular. If we consider a model in the form of a graph or a sequence, Keras allows us to save the model we are working on to be used in other applications using the save () method.
- Predefined datasets: Keras contains large predefined datasets. We can use this dataset to import and upload it directly (example: IMDB DATA contains approximately 25,000 movie reviews). This dataset contains binary numbers (0 and 1) to review each movie. 0 represents the negative sentiment and 1 represents the positive feeling. We can upload IMDB DATA as:

```
from keras.datasets import imdb
(x_train, y_train), (x_test, y_test) = imdb.load_data()
```

- Assessment and prediction: Keras contain the methods evaluated() and predict(). These methods can use the NumPy dataset. After testing the data, the result is evaluated.
- Pre-trained models: Keras contains a number of pre-trained models. These templates can be imported from keras.applications. They are useful for extracting and fine-tuning of characteristics. Keras.application is a module that contains weights (model weights are all the parameters including trainable and non-trainable of the model which are in turn all the parameters used in the layers of the model) for image classification such as VGG16, VGG19, Xception etc.
- Layers in Keras: There are many layers and parameters in Keras. All Keras layers contain a number of methods. These layers are useful for building, training, configuring data. The dense layer is beneficial for the implementation of operations. "Flatten" is used to reduce the flow of data input. The input is used to initiate a Keras tensor.

```
keras.layers.Reshape(target_shape)
kera.layers.Flatten(data_format=none)
keras.layers.Dropout(rate, shape_noise=none, seed=none)
```

- We can get the result of an intermediate layer: Keras has an essential advantage when used in ML models allowing us to view the result in the middle of a layer. To get intermediate results, we can simply create a new layer in which to store the viewed result.
- Keras is the native Python library: It uses all the known concepts from Python so that the knowledge of this programming language makes the development of an ML model in Keras very simple.
- Data preprocessing: Keras offers us several functions for data preprocessing. ImageDataGenerator is one such method. It can be imported by:

```
from keras.preprocessing.image import ImageDataGenerator
```

This function helps resizing images, changing the degree of tilt - rotation, and changing the height and width of the image.

2.3.2. Keras How does it works?

The diagram below shows the main steps in building a model in Keras (Fig. 3):

- Defining a neural network: in this step, we define the different layers in our model and the connections between them. Keras has two main types of models: sequential models and functional models. We choose the type of model we want and then define the data flow between them.

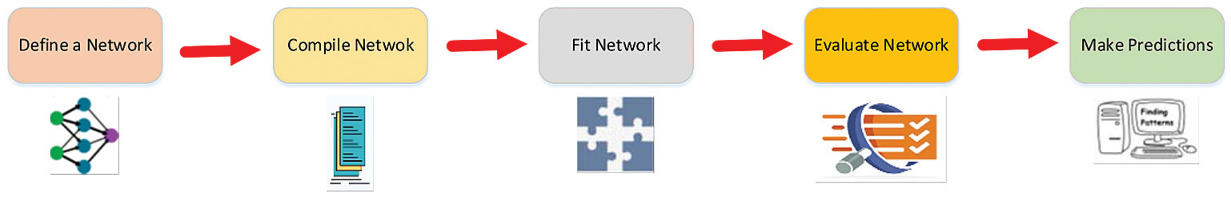

Fig. 3. Building a model in Keras.

- Compiling the network: compiling a sequence of code means converting it into a suitable form for the machine to understand. In Keras, the model.compile () method performs this function. To compile the model, we define the loss function that calculates the losses in our model, the optimizer that reduces the loss, and the measurement that is used to find the accuracy of our model.
- Network adaptation: in this stage we integrate the model to our data after compilation. It is used to train the model on our data.
- Network evaluation: After adapting our model, we need to evaluate the errors
- Making predictions: we use model.predict() to make predictions using our model on new data, different from those on which it was trained.

2.3.3. Advantages Keras

- Simplicity: Keras is very simple and easy to use. It is an easy-to-use API with an easy-to-learn and coding function. It is very easy to learn the basics of Deep Learning using Keras. The functions in Keras are very simple making it easy to design the desired neural network models.
- Backend support: Keras does not work with low level computational models - it works as an interface for TensorFlow, Theano and Microsoft CNTK, these are just a few libraries that Keras uses for backend support - this gives the user the opportunity to choose a compliant backend requirement.
- Pre-learned models: Keras offers many pre-learned models, which help users to simplify their tasks. These models allow the user to make fine adjustments, feature extraction and prediction calculations.
- Fast experimentation: Keras is built to simplify user tasks. Therefore, it has the ability to build neural network models using as few lines of code as possible. It provides fairly good support for features that allow users to deploy models quickly.
- Extensive user community and "calibre" documentation: Keras has a large user community and is an open-source platform. This community allows researchers to publish their code and experiment details to be useful for other users. It usually answers all questions asked by users. The "calibre" documentation provides easy-to-use support for installing and configuring Keras and contains every detail of the features documented as well as tutorials for using them.

2.3.4. Disadvantages Keras

- Uncomplete features: There are some features in Keras for which there is room for improvement. It lacks some types of pre-learned models for use. Keras does not support dynamic chart creation functions.
- Inefficient error management: errors given by the Keras library do not provide efficient error management. There is a need to make errors easier to identify. It is quite complicated to find the main cause in the occurrence of an error, debugging is difficult to perform.
- Low level API: Keras generates low level errors quite often. The reason for this is that there are certain low-level features and operations that Keras is not capable of running as a front end for TensorFlow, Theano and Microsoft CNTK.

2.4. PyTorch

PyTorch is an open-source machine learning system based on the Torch library used for applications such as computer vision and language processing developed mainly by Meta AI. It is a free and open-source software released under the modified BSD license. Although the Python interface is more refined and is the main point of development, PyTorch also has a C ++ interface (Wikipedia, "Pytorch")

A number of deep learning programs are built on PyTorch, including Tesla Autopilot, Uber's Pyro, Hugging Face's Transformers, PyTorch Lightning and Catalyst.

2.4.1. PyTorch Main Features

The two main features of PyTorch are:

- Tensor calculation (similar to NumPy) with consolidated support for GPU (Graphical Processing Unit) acceleration
- Automatic differentiation for the creation and training of deep neural networks

In PyTorch, modules are used to represent neural networks.

These are presented below.

1. Autograd

The autograde module is PyTorch's automatic differentiation engine, which helps to calculate gradients quickly.

2. Optim

The Optim module is a package of pre-written optimization algorithms used to build neural networks.

3. nn

The nn module includes various classes that help build neural network models. All modules in PyTorch are subclasses of the module nn.

Features of PyTorch workflow:

a. Dynamic calculation graph in Pytorch:

The calculation graphs in PyTorch allow the framework to calculate the gradient values for the constructed neural networks. PyTorch uses dynamic graphs. The graph is defined indirectly using operator overload. Dynamic graphs are more flexible than static graphs, in which users can make interleaved constructions and evaluate graphs. These are easy to troubleshoot because they allow line-by-line code execution. Finding problems in model code is much easier with PyTorch dynamic graphics - an important feature that makes PyTorch a popular choice in the industry.

PyTorch computational graphs are rebuilt from scratch at each iteration, allowing the use of random Python control flow statements, which can affect the overall shape and size of the graph each time an iteration occurs.

b. Uploading data

Working with large data sets requires that all data be loaded into memory in a single operation. This causes the memory to overload and the programs to run slowly. In addition, it is difficult to keep the data sample processing code. PyTorch offers two data primitives - DataLoader and Dataset - to parallelize data loading with automatic batches and better code readability and modularity. DataSet and DataLoader allow users to use their own data as well as preloaded data sets. While Dataset hosts those samples and labels, DataLoader combines the dataset and sampler by deploying an iteration around the dataset so that users can easily access predefined examples.

2.4.2. PyTorch How does it works?

PyTorch uses a technique called automatic differentiation that numerically evaluates the derivative of a function. Automatic differentiation calculates back passes in neural networks. In neural network training, weights are randomly initialized to numbers that are close to zero, but not zero. Shifting backwards is the process by which these weights are adjusted from right to left, and a shift forward is reversed (from left to right).

torch.autograd is the library that supports automatic differentiation in PyTorch. The main class of this package is torch.Tensor. To track all operations on it, we must set. requires_grad to True. To calculate all gradients, we must call. backward () method. The gradient for this tensor will be accumulated in the .grad attribute.

If we want to detach a tensor from the calculation history, we call the .detach () function. This will also prevent future calculations on the tensor from being tracked. Another way to prevent history tracking is by wrapping the code with torch.no_grad ():

The Tensor and Function classes are interconnected to build an acyclic graph that encodes a complete calculation history. The grad_fn attribute of the tensor refers to the function that created the tensor. To calculate derivatives, we must call.backward() on a tensor. If the tensor contains a element we do not need to specify any parameters for the back function. If the tensor contains more than one element, we specify a gradient that is a tensor of a shape appropriate to the needs of the operations to be performed.

For example, we create two tensors, one with requires_ grad as True and the other as False. We will then use these two tensors to perform addition and summation operations. After that, we will calculate the gradient of one of the tensors:

```
a = torch.tensor([3.0, 2.0], requires_grad=True)
b = torch.tensor([4.0, 7.0])
ab_sum = a + b
ab_sum
ab_res = (ab_sum*8).sum()
ab_res.backward()
ab_res
a.grad
```

The return value of calling. grad on b is null because we did not set its requirements_ grad to True.

2.4.3. Advantages PyTorch

The main difference between PyTorch and other models, such as TensorFlow, is the support for dynamic neural networks. TensorFlow treats the neural network as a static object; if we want to change the behavior of the model we have to start from scratch. With PyTorch, the neural network can be adjusted on the fly while running, making it easier to optimize the model.

Another major difference is the way the code is debugged. Effective troubleshooting with TensorFlow requires a special tool that allows us to examine how network nodes do their calculations at every step. PyTorch can be debugged using one of the many widely available Python debugging tools.

The last major advantage of PyTorch is the ease with which it can distribute computing work across multiple CPU or GPU cores. Although this parallelism can be achieved in other machine learning tools, PyTorch is much easier to operate from this point of view.

2.4.4. Disadvantages PyTorch

Despite its advantages, PyTorch has some shortcomings. An official version 1.0 has not yet been released, so it is not stable enough for production work, while TensorFlow and other similar models have more official variants and therefore better support, more detailed documentation and communities. of older users. TensorFlow also comes with Tensorboard, a highly capable visualization tool for building the model graph (a "map" of the neural network) and various data representations that specialize in machine learning. PyTorch does not have that yet, so developers will need to rely on one of the many existing tools for viewing Python data.

3. CONCLUSIONS

The analysis made on the most popular and advanced libraries for neural networks reveal that the best candidates for a ML model could be in fact a combination of two: Keras and TensorFlow that can be used with succes to build a neural network for the automatic intelligent recognition of microfossils.

TensorFlow is an end-to-end open-source platform, a library for multiple machine learning tasks, while Keras is a high-level neural network library running on TensorFlow. Both provide high-level APIs used to easily build and shape models, but Keras is easier to use because it has built-in Python.

TensorFlow is generally used when working with large data sets used to detect objects with excellent success rates and high performance. TensorFlow runs on Linux, MacOS, Windows and Android. The framework was developed by Google Brain and is currently used for Google's research and production needs.

Keras works as a cover for the TensorFlow frame. Thus, we can define a model with the Keras interface, which is easier to use, then we will access TensorFlow when we need to use a feature that Keras does not have or we want to use specific TensorFlow features by placing the TensorFlow code directly in the Keras training model.

ACKNOWLEDGMENTS

The author wishes to express his special thanks to Dr. Mihaela Melinte-Dobrinescu for providing literature and useful comments on an earlier version of the manuscript. Also, I wish to express my gratitude to Dr. Eliza Anton for helping review this article before its final form.

The financial support of this paper was provided by the Ministry of Research, Innovation and Digitization through Program 1 – PN III – Research of Excellence, Project PFE 23/30.12.2022 AMBIACVA.

REFERENCES

Brasier, M.D. (1980). Microfossils. Brief overview of major groups of microfossils. Simple drawings of each kind of microfossil. *George Allen & Unwin*, *London,* 193 p.

Cui, X., Qiao, T., Zhu, M. (2019). Scale morphology and squamation pattern of Guiyu oneiros provide new insights into early osteichthyan body plan. *Scientific Reports*, **9**(*1*): 4411-4423.

Ehrenberg, C.G. (1836). Bemerkungen ueber festemikroskopische anorganische Formen in den erdigen undderben Mineralien. *Königl Press. Akad. Wiss. Berlin*, 84-85.

Fawad Malik (2021). Top 5 Open Source AI Image Processing Solutions; Webtech Solution Blog.

Haq, B. U., Boersma, A., Eds. (1978). Introduction to Marine Microfossils. *Elsevier New York*, 376 p.

Sydorenko, I. (2021). What Is TensorFlow? Review, Definition, Usage, & Implementation; LaberYourData.

Melinte, M. C. (2004). Calcareous nannoplankton, a tool to assign environmental changes". *Geo-Eco-Marina*, **9-10**: 17-25.

Brocklehurst, R.J., Schachner, E.R., Sellers, W.I. (2018). Vertebral morphometrics and lung structure in non-avian dinosaurs. *Roy. Soc. Open Sci.* **5**(*10*): Art. No. 180983.

Saraswati, P.K., Srinivasan, M. (2016). Micropaleontology: Principles and Applications. *Springer Switzerland*, 223 p.

Tappan, H. (1980). Haptophyta, Coccolithophores and other calcareous nannoplankton. *The Paleobiology of Plant Protista. Freeman San Francisco*, 678-803.

Thierstein, H.R., Young, J.R. (2004). Coccolithophores: from Molecular Processes to Global Impact. *Springer Berlin Heidelberg*, 565 pp.

www.wikipedia.org/wiki/PyTorch – Wikipedia "Pytorch"

www.opencv.org/about – "About OpenCV"

www.tensorflow.org – "Why TensorFlow"