



Mindtree

A Larsen & Toubro Group Company



Data science and applied AI: Mindtree viewpoint on **ML Frameworks.**

Comparing ML frameworks



In the process of simplifying life, artificial intelligence is gaining immense popularity. Many industries are embracing AI to improve efficiency, productivity, revenue, and deliver superior experiences. With newer deep learning models, AI can gather and analyze large quantities of structured or unstructured data in the form of tables, text, or images—generating valuable business insights. Businesses on their path of adopting AI are often faced with a multitude of challenges like:

- How would the business benefit from AI?
- When to adopt AI?
- How and where to start?
- Which frameworks should we use?

In this POV, Anand Sridhar Rao, General Manager, Data and Intelligence, Sylvester Daniel John, Head of Applied AI Center of Excellence, and Samson Saju, Senior Research Engineer, Mindtree, share their views on the three prominent deep learning frameworks, namely, **PyTorch, Keras, and TensorFlow**. They also study their suitability for different applications.

TensorFlow

TensorFlow is a symbolic programmatic library developed at Google Brain for research in symbolic programming. In the year 2015, it was made open source. TensorFlow is developed using C++ and Python. It is designed around dataflow graphs. Data flow graphs define computations mathematically using a directed graph. The nodes on the dataflow graph represent computations (mathematical operations on Tensors), while edges define how the data (Tensors) moves through the graph. Since the data flowing through these graphs are tensors, the framework was aptly named TensorFlow. The edges in the dataflow graph are directed to connect the source and sink computation nodes. TensorFlow enables low-level hardware support using CUDA on GPUS and also supports Tensor Processing Units (TPUs.)



PyTorch

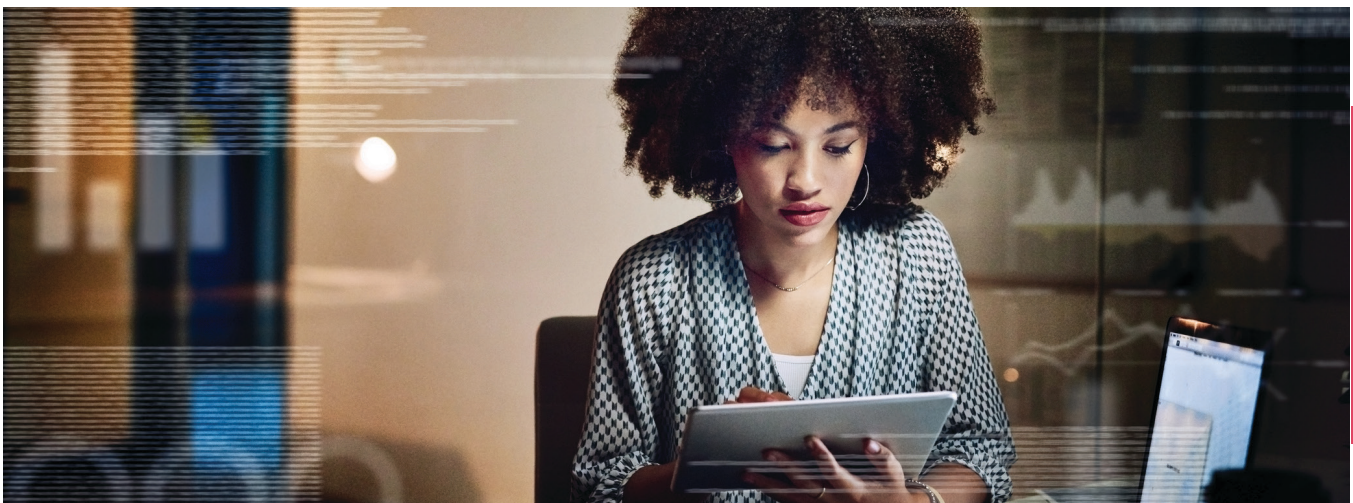
PyTorch project is an evolution of Torch, a C-based tensor library with Lua. It was made open source by Facebook in 2017 and is written in Python. PyTorch facilitates experimentation and rapid prototyping as it allows operations to query code without a full model build. PyTorch has a very Pythonic way of doing things, making it friendly to developers migrating from NumPy, as it offers a practically identical set of abstractions combined with GPU acceleration.

“I chose PyTorch for my Deep learning experiments when it was in its initial days. I haven't looked back after that. Before that, I worked with many deep learning frameworks like Caffe, Theano, Keras, TF, etc. But none of them was pure python framework; either there was heavy configuration involved or learning specific DSL(Domain Specific Language) or both. With PyTorch, you can write all your code in python, and the framework takes care of making sure it runs on CUDA with a comparable speed of execution. This will help you think directly in the language you're programming rather than working your way through the complicated python program and DSL and their interaction. Also, easy to understand error messages and to debug like normal python program.”

Manish Patel, Principal Research Engineer, Mindtree.

Keras

Keras was originated by AI researcher François Chollet in 2015, shortly before joining Google as a deep learning researcher and engineer. It was developed as a 'beginner-friendly' high-level API for complex and powerful deep learning frameworks. It is written in Python and is available under an MIT license with Python, R, and now JavaScript interfaces. However, as of June 2020, the latest version of Keras (2.3.0) has been announced as the last version that would support secondary platforms beyond TensorFlow.



Trends

To validate the adoption of these frameworks under various settings like Academia and industry, we looked at various sources of public data. Primarily we try to identify the popularity of these frameworks in research and production.

Research trends

To study the popularity of these frameworks in the research community, we analyzed data from popular ML conferences like NIPS, CVPR, etc. We looked at data from 2016 onwards on TensorFlow and PyTorch and observed that over the years, the number of papers using PyTorch for their implementations has been steadily rising and currently hovering above ~70%. We also observe a decline in the use of TensorFlow starting from 2018. The solid lines in the graph below illustrates the number of papers that use Pytorch, while the dotted lines represent the number of papers that use Tensorflow.

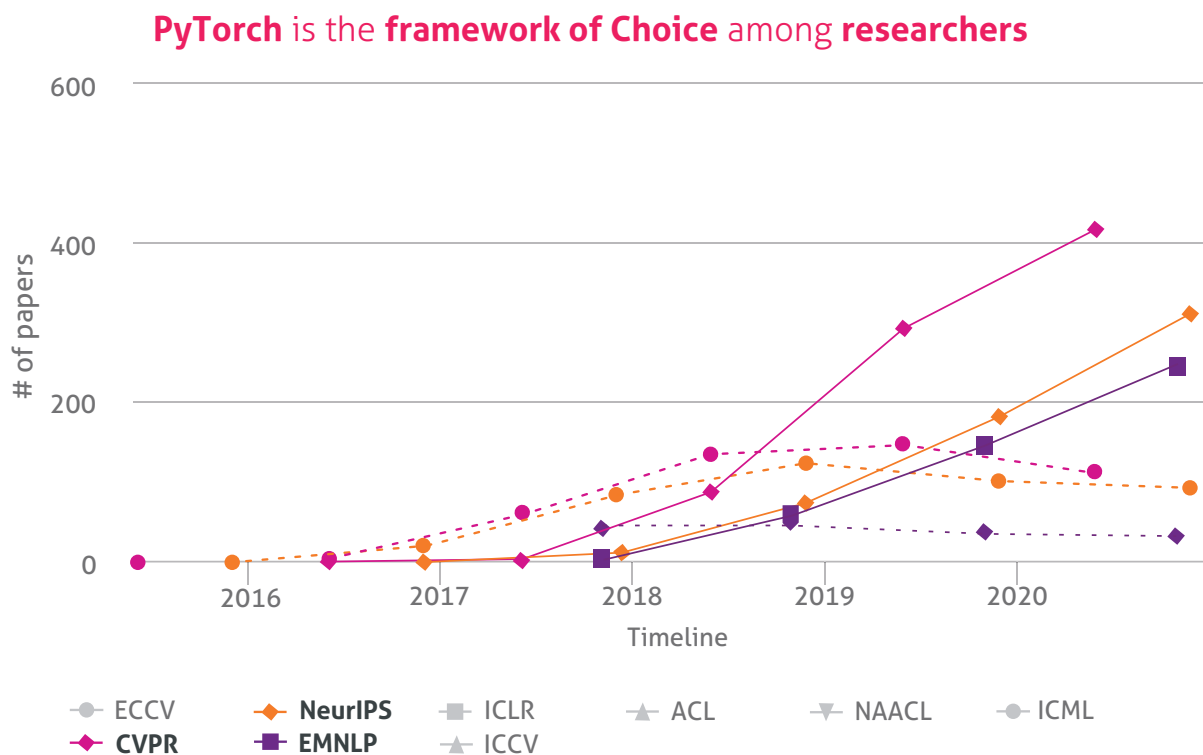


Exhibit 1: Raw counts of papers using PyTorch (Solid) vs TensorFlow(Dotted) in prominent AI conference
(Source:<http://horace.io/pytorch-vs-tensorflow/>)



“My work area is mostly targeted towards research and exploration. As part of my projects in Mindtree, my work is to implement a research paper, which requires BERT modification. For this project, we chose PyTorch, and the implementation was seamless. We picked up PyTorch because of its ease of usability, better debugging (due to dynamic computational graph, a contrast to static computational graph in TensorFlow), and a plethora of learning material available online.”

Utkarsh Pratiush, Research Engineer, Bangalore.

Pre-trained model availability

New models are often built by looking at the prior art and identifying good candidate models related to the task at hand. This greatly aids model building by reducing computational costs and leveraging benefits from transfer learning. We looked at the availability of pre-trained models as a key parameter in the success of an ML framework. We analyzed public data from the website papers with a code on the availability of pre-trained models on various ML frameworks. We observed a trend similar to the one seen in the research community, showing an increase in PyTorch-based implementations and a reduction in TensorFlow-based implementations. We observed that the number of implementations for PyTorch has doubled since 2018, while this number has halved for TensorFlow. We also observed that the number of implementations in other languages is also steadily declining. The data visualization depicting this is shown in the below graph.

March 2018	
PyTorch	25%
TensorFlow	32%
Others	41%

March 2021	
PyTorch	54%
TensorFlow	16%
Others	31%

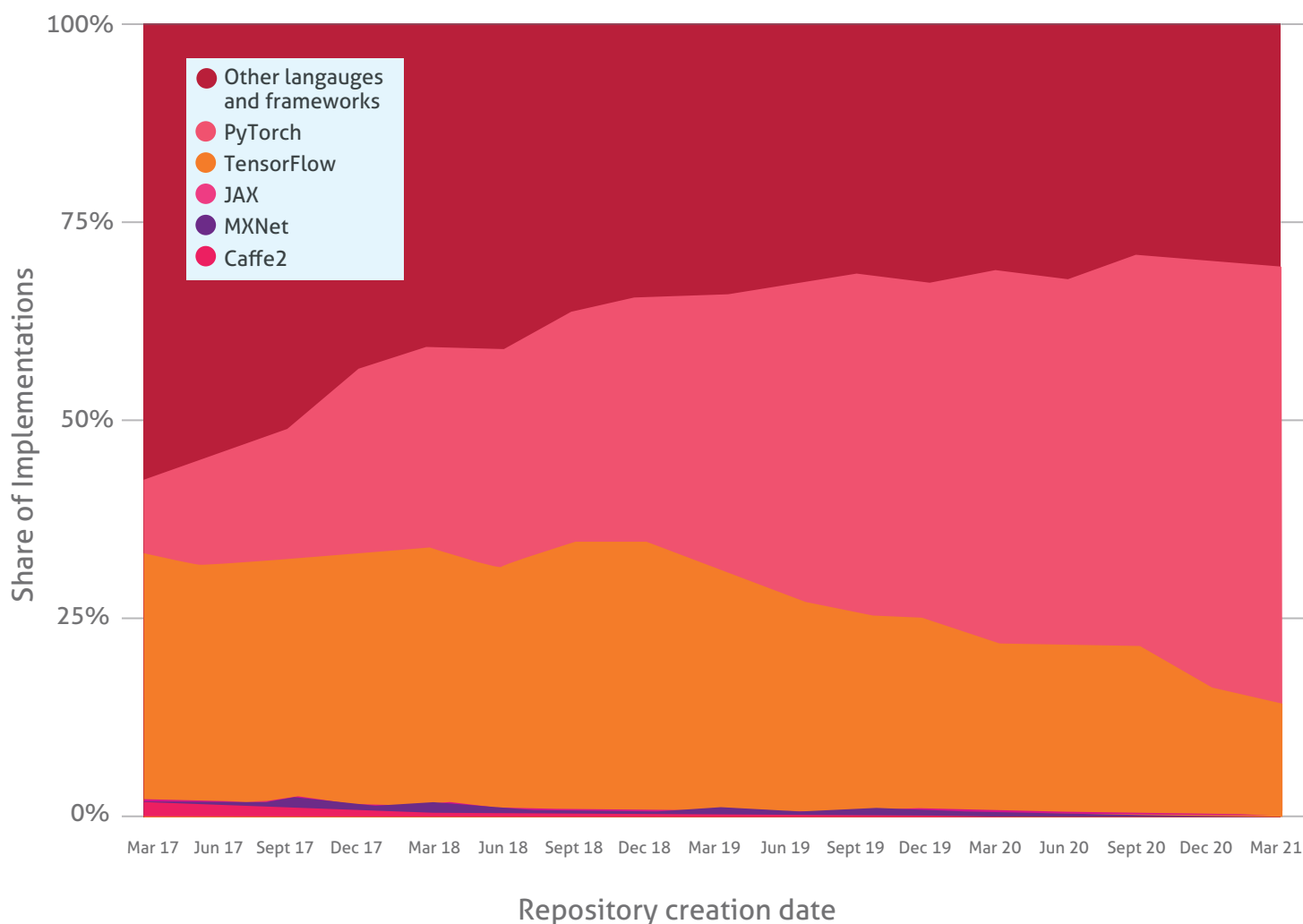


Exhibit 2: Availability of pre-trained models in various ML frameworks
(Source: <https://paperswithcode.com/trends>)

Industry trends

To study the popularity of various ML frameworks in the industry, we used statistics around job postings in Europe. We looked at the job postings on indeed.com as a proxy for measuring the adoption of various ML frameworks in the industry. Based on our analysis, we noticed that TensorFlow has roughly two times more job openings than PyTorch. We also observed that PyTorch had significantly more job postings than Keras in most geographies. In Europe, Germany has the highest job postings for AI, followed by UK and France. In these countries, TensorFlow has roughly two times more job openings than PyTorch, and PyTorch has roughly two times more job openings than Keras. Therefore, we concluded that, currently, TensorFlow is the ML framework of choice within the industry.

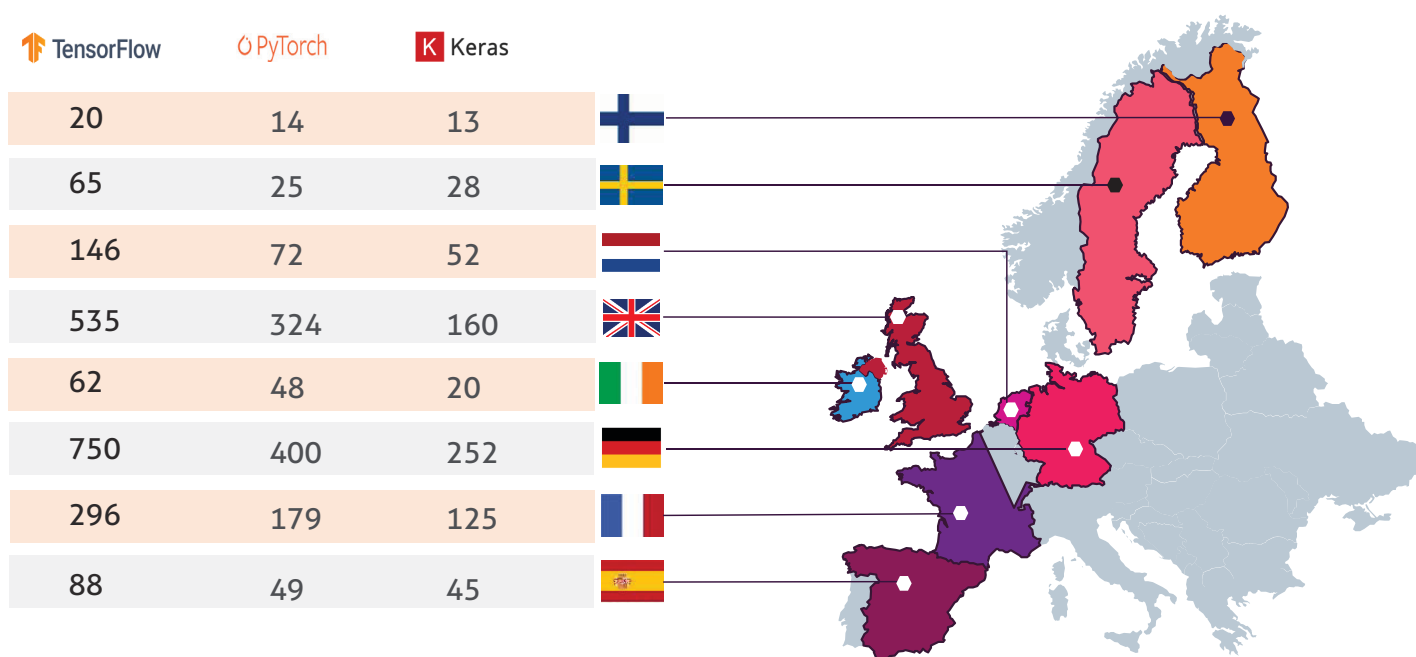


Exhibit 3: Number of Job openings in Indeed for various ML frameworks across Europe (Source: Indeed)



"My favorite ML Framework is TensorFlow as it has better support for various production settings with support for various types of computes like CPU, GPU, TPU."

Josyula Chandra Shekar, Senior Manager, Mindtree.

Conclusion

Our studies indicate that PyTorch is the framework of choice among researchers, and TensorFlow is currently popular among industries. However, as newer and efficient pre-trained models become available on PyTorch, we expect to see an increase in the adoption of PyTorch within the industry. As more Ph.D.'s and graduates migrate from Academia to industry, their preferences would turn the tide in favor of PyTorch within the industry.

Feature comparison

Introduction

In this section, we compare the features of TensorFlow, PyTorch, Keras under two settings, namely research and production. In the research setting, we emphasize features like rapid prototyping, easy debugging, good documentation, and availability of SOTA pre-trained models. In the production, setting importance is given to speed, ease of deployment, no Python overhead, support for mobile and edge devices.

ML frameworks for research

As data showed us that PyTorch is the framework of choice in the research community, we asked the question: what makes PyTorch popular with researchers?

PyTorch is very Pythonic and is similar to Numpy. PyTorch has a well-designed and documented API aiding in faster prototyping, while TensorFlow's API has become very confusing due to multiple API changes like 'layers' -> 'slim' -> 'estimators' -> 'tf.keras'. Being very Pythonic, PyTorch integrates well with the Python ecosystem and supports PDB (Python debugger). Another favorite feature of PyTorch is its ability to query code without a full model build.

TensorFlow and Keras for research

TensorFlow has recently reached a function-wise parity with PyTorch with its new TensorFlow 2.0 update. However, since Pytorch has already reached most of the research community, migration back to TensorFlow is likely to be slow due to several factors.

As PyTorch implementations are easier to come by, authors are incentivized to publish their code in PyTorch for improved acceptance within the community. Since researchers often work by improving existing state-of-the-art models, it is to be noted that major research teams like Google/Deepmind continue to use TensorFlow.

TensorFlow recently introduced TensorFlow eager mode, which is almost identical to PyTorch's eager mode, which was popularized by Chainer. This enables TensorFlow to have almost all of the advantages of PyTorch's eager mode (developer friendliness, debugging, etc.) Some disadvantages of TensorFlow eager models are that these models would not work in a non-Python environment, mobile devices, etc. Keras being a high-level API, often fails to deliver easy access/manipulation of the underlying API for complex tasks like custom layer creation.

ML frameworks for production

Industries focus on production, and production needs greatly vary based on the research needs. TensorFlow was built to specifically address requirements like no Python and mobile. Hence, TensorFlow has out-of-the-box solutions for all of these issues. TensorFlow's design choice of using data flow graphs and execution engine natively does not need Python. Similarly, TensorFlow Lite and tensor flow serving are specifically designed to address mobile and serving requirements. Since its inception, PyTorch was designed to cater to research needs and has fallen short in terms of features catering to production. And as a result of this, many organizations currently use TensorFlow in production.

But recently, PyTorch has come up with solutions to most production requirements, like PyTorch TorchScript for model deployment in C++ without Python dependency, TorchServe for serving, and PyTorch Mobile for mobile. PyTorch has introduced a new “graph” representation for computing named TorchScript. A PyTorch model can be converted to a Torch script using two techniques tracing and script mode. Tracing works by recording the operations executed by the function while processing the input and using the recorded operations to build an IR (Intermediate Representation) that works with TorchScript.

One major limitation of tracing is its inability to capture operations that were not executed while processing the input (e.g., the if or else blocks of code in the input function). The alternative option to building a TorchScript is to use the script mode, which tries to build the IR (Intermediate Representation) by reinterpreting the Pythonclass/function. Converting PyTorch models to TorchScript gives us the benefits like no Python dependency, which becomes critical for certain production scenarios.

Feature comparison chart

 TensorFlow

 PyTorch

 Keras

Programming API

- Low Level API
- Steep learning curve
- Confusing as APIs switched many times (e.g. 'layers' -> 'slim' -> 'estimators' -> 'tf.keras')

- Low Level Api
- Medium learning curve
- Very Pythonic similar to NumPy
- Very flexible easy to write custom layers
- fast.ai, Flare, and Ignite serves as high level api

- High level abstracted API for the low-level functionality of TensorFlow, with the capability to create six types of core layers (input object, dense layer, activation layer, embedding layer, masking layer, and lambda layer.)
- Hard to create custom layers.
- Mixed learning curve easy to start hard to mature.

<p>Accessibility and debugging</p>	<ul style="list-style-type: none"> • Reference GitHub implementations easily available for projects from Google and Deepmind but harder to come by for papers from other sources • TensorFlow debugging needs active session • TensorFlow eager improved the debugging, eager models can't be exported to non-python environment • Offers a dedicated debugging module for debugging which is usually overwhelming for users • TensorFlow 2.0 with eager execution has debugging better recently 	<ul style="list-style-type: none"> • Reference GitHub implementations for newer papers are easier to find as more researchers are adopting PyTorch • Debugging is easy • PyTorch supports standard python debugger pydb, making debugging easier. • PyTorch allows operations to query code without a full model build. 	<ul style="list-style-type: none"> • Reference GitHub implementations for newer and or complex papers are harder to find • Underlying code more difficult to navigate than in PyTorch • Harder to identify the point in the dependency chain at which the code is causing problems • Keras is easy to work, but since it has too many levels of abstractions on backend frameworks, debugging gets tricky. • Should have understanding of the lower-level API i.e. mostly TensorFlow for debugging
<p>Prototyping</p>	<ul style="list-style-type: none"> • TensorFlow prototyping is more harder owing to more complex api and harder debugging 	<ul style="list-style-type: none"> • PyTorch easy prototyping as it is very pythonic and similar to NumPy. • Lot of reference material and sample implementations makes prototyping fast 	<ul style="list-style-type: none"> • Keras being a high level api support fast prototyping when using standard neural network layers. • Prototyping becomes cumbersome if custom layers or complex architectures are to be built as this would require access to low level api and the high level abstraction of Keras makes it more complex
<p>Beginner friendliness and documentation</p>	<ul style="list-style-type: none"> • Daunting for beginners • Multiple switches (e.g. 'layers' -> 'slim' -> 'estimators' -> 'tf.keras') have made the documentation more confusing. • Being a symbolic programmatic library built around dataflow graphs makes it harder to learn and daunting for beginners. 	<ul style="list-style-type: none"> • More complex than Keras but easier than TensorFlow • PyTorch is less hindered by the outdated 'authority' posts that can plague the Keras initiate, since PyTorch's development has been more consistent since its inception. • PyTorch maintains an active and helpful user forums 	<ul style="list-style-type: none"> • Considerably easy for beginners. • Keras leans on the Stack Overflow community. • Keras-related documentation is lacking in practical solutions for some common problems, or an adequate number of code examples needed for the breadth and scope of developers' issues. • Keras developer information is found in discrete communities on Discord or other gated channels, hidden from search indexes.




Supported languages	Python, JavaScript, C++, Java, GO, Swift.	Python, C++, and Java.	Python, R
Dataset size	<ul style="list-style-type: none"> • Suitable for working with large datasets • Provides data structures to work with various types of data 	<ul style="list-style-type: none"> • Suitable for working with large datasets • Provides data structures to work with various types of data 	<ul style="list-style-type: none"> • Built for quick prototyping is slower. • Less suitable for large datasets.
Speed	Fast and supports both CPU and GPU However, rumored to be slightly slower than PyTorch	Fast and supports both CPU and GPU However, rumored to be slightly faster than TensorFlow	Slower than TensorFlow and PyTorch being a high level api
Industry adoption	<ul style="list-style-type: none"> • Widely adopted in the industry (based on job postings) • Matured solutions TensorFlow lite and TensorFlow serving for deployments 	<ul style="list-style-type: none"> • Picking up in industry adoption (based on job postings) • Features like Torchserve, PyTorch mobile catering to industry needs 	<ul style="list-style-type: none"> • Less popular in research community • Being a high level api harder to create custom layers
Research adoption	<ul style="list-style-type: none"> • Losing popularity among researchers • Less pythonic • Poor documentation , harder debugging fewer reference implementations. 	<ul style="list-style-type: none"> • Most preferred in the research community. • More pythonic • Easy to debug and lot of reference implementations on GitHub 	<ul style="list-style-type: none"> • Less popular in research community • Being a high level api harder to create custom layers
Deployment/ production	<ul style="list-style-type: none"> • TF built around production needs like No python overhead ,Mobile, efficient serving etc. • TensorFlow Serving is a mature package for custom model deployments. • Supports TPU 	<ul style="list-style-type: none"> • PyTorch JIT intermediate representation (IR) of PyTorch called torch script allows model deployment in C++ without python dependency (relatively new) • Torch script has no impact on standard PyTorch code. • TorchServe package reduces need for custom code for model deployments 	<ul style="list-style-type: none"> • Keras has a mature ecosystem of packages for speed deployments. Owing to its lower level api being tensorflow. • Supports multiple backends like Tensorflow, Theano, CNTK etc • Similar support as Tensorflow when using tensorflow as back end

Mobile and EDGE	<ul style="list-style-type: none"> • TensorFlow Lite • Supports Android, Linux and IOS • Supports Python free deployment • Supports ARM Cortex-M series, ESP32 • Arduino library available, supports multiple development boards • All TensorFlow operations are not supported 	<ul style="list-style-type: none"> • PyTorch Mobile (Beta) • Supports Android, Linux and IOS • Supports ARM CPU's • Supports Quantization • Supports various backends like GPU,NPU, DSP available soon in Beta 	<ul style="list-style-type: none"> • Models are ported to TensorFlow lite • Will have to use low level APIs • Limitations of TensorFlow lite api
Used by companies	Google, AirBnB, AMD, Bloomberg, LinkedIn, PayPal, Qualcomm, Snapchat	Facebook, Genentech, JPMorgan Chase, Microsoft, Salesforce, Toyota, Wells Fargo	Apple, Google, Nvidia, Microsoft, Netflix, Uber, Amazon AWS

Exhibit 4: Feature comparison chart (Our preference is highlighted)

When and which?

It's hard to pick a winner among the three ML frameworks, and it often comes down to the use case. We recommend TensorFlow for scenarios that focus on multi-platform support, IoT, EDGE, Mobile as offerings around these are more mature in comparison to PyTorch. We recommend PyTorch for most applications due to its Pythonic API, availability of SOTA pre-trained models, and ease of creating complex network architectures. We don't advise using Keras as it is primarily developed for rapid prototyping. It often becomes challenging to make custom layer modifications in Keras owing to its abstraction over the underlying framework. However, Keras can be used as a tool for rapid prototyping of TensorFlow models and eases the steep learning curve of TensorFlow.

 TensorFlow <ul style="list-style-type: none"> • Multi-platform support IOT, EDGE, Mobile etc. • High performance • Functionality 	 PyTorch <ul style="list-style-type: none"> • Flexibility • Research • High performance • Debugging capabilities • First release advantage 	 Keras <ul style="list-style-type: none"> • Rapid prototyping • Entry point for TensorFlow • Small dataset • Multiple back-end support
--	---	--

“I like Keras owing to its ease of use. In my line of work, I need to rapidly prototype deep learning models for various classification tasks and the ease to build models in Keras comes in handy for me.”
Raja Ray, Technical Architect, Mindtree.

“Keras is my framework of choice because of its better support for EDGE devices, and also it simplifies the APIs for TensorFlow substantially reducing the learning curve and delivers better developer experience.”
Maheswaran Venkatramani, Associate Tech Lead, Mindtree NXT.

Future of ML frameworks

With over 8.2 million Python developers, Python has been the language of choice for machine learning and deep learning, which is 1 million more than Java and over 6 million more than Swift. One of the major reasons for the higher adoption of Python is its simplicity and ease of use. However, one of Python's major drawbacks is its execution speed; it is 400 times slower than C++. In machine learning, we get around this by using the libraries written in more efficient languages like C. Also, for many production scenarios, the overhead of Python is unacceptable and existing frameworks are trying to address this with TensorFlow serving and PyTorch JIT. A lot of effort is being put into improving the performance of ML frameworks. For example, JAX, a Python library that is a direct successor of the Autograd library, improves computation speed. And, Julia, an alternative language to Python, tailor-made for the ML/AI community, has the simplicity of Python and boasts a performance similar to that of C language.

Python, TensorFlow, and PyTorch have been with us for so long. Julia is a relatively new framework, and we are yet to recognize its full potential yet. Julia does show some promise and is well on its way to being one of the best languages for data computations. Julia manages to be faster than Python while being as easy to use as Python. However, some questions for which the answer lies in the future are, How will Python, PyTorch, and TensorFlow evolve to address the increasing demand for computational speed? How fast will the developer community adopt Julia? How will Julia evolve next? I guess we will need to wait and watch.



"Julia is turning out to be the future of AI frameworks mainly because of:

- 1) Easy to use like python but speeds comparable to C.*
- 2) With scientific machine learning evolving as a field, which essentially combines neural networks with traditional scientific computing methods. The resulting model works very well even though trained on less data. Julia is particularly targeted towards such applications.*
- 3) Julia's code is easily readable despite complicated equations coded in it. This results in fewer challenges in terms of communicating a complex idea among the collaborators. Thus, it saves a lot of time."*
- 4) Has a plethora of libraries, hardware support, and community backup. The Juliacon held every year provides a platform for researchers to showcase their work and bring about a promising learning environment for everyone."*

Utkarsh Pratiush, Research Engineer, Mindtree.



Conclusion

AI has started yielding fruits to its early adopters and is one of the major investment areas for many businesses. However, AI transformation is a slow process of gathering the right data and gradually building complex AI systems powered by state-of-the-art deep learning models to unlock new opportunities, improve operational efficiency, and generate quality business insights. The AI transformation of a business has to be carefully planned and executed. We at Mindtree strive to make AI pervasive for your business. **Welcome to Possible.**

Sources

- [getting started:training and prediction with keras](#)
- [Kears for R](#)
- [Tensorflow documentation](#)
- [Keras Documentation](#)
- [Pytorch Documentation](#)
- [IFlexion](#)
- [The Gradient](#)
- [MLK making AI simple](#)
- [Tensorflow vs Keras vs Pytorch: Which Framework is the Best?](#)
- [Papers with code](#)
- [Indeed](#)
- [Horace data Viz](#)
- [Julia Vs Python](#)
- [JAX](#)
- [Julia](#)

The authors would like to thank the following Subject Matter Experts from Mindtree for their participation in the survey conducted for this POV:

Raja Ray, Kishore Kumar Malik, Meenakshi Annamalai, Arish Balasubramani, Utkarsh Pratiush , Ashraf Ali, Josyula Chandra Shekar, Suvesh Pattnak, Ramprasad MK, Maheswaran Venkatramani, Kailash Hari SN, Nikita Pise, Ektha Mallya, Seemant Singh, Pravin Bhanjantri, Aman Maurya, Geetha Sreenivasan, Sasikiran Karri, and Manish Kumar Patel.

Meet the authors



Samson Saju

Senior Research Engineer, Mindtree.

<https://www.linkedin.com/in/samson-saju/>

Samson has over five years of experience working in various AI and deep learning technologies in the area of NLP and computer vision. He was the Mindtree Stanford Research Fellow for the year 2019-2020.



Anand Rao

General Manager, Data and Intelligence Advisory, Mindtree.

<https://www.linkedin.com/in/anandrao81/>

Anand is responsible for outlining the GTM offerings and capabilities and work with enterprises in conceptualising and accelerating practical cloud-led, data-driven cognitive solutions for business problem statements, and guide the strategy and delivery of vibrant AI and ESG Tech programs and ecosystem.

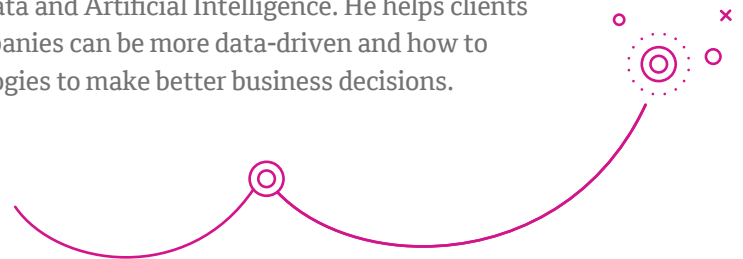


Sylvester Daniel John

Head of Applied AI Center of Excellence, Mindtree.

<https://www.linkedin.com/in/sylvesterdj/>

Sylvester is a Senior Big Data Architect around 17 years of IT experience, covering the areas of Big Data and Artificial Intelligence. He helps clients understand how their companies can be more data-driven and how to leverage emerging technologies to make better business decisions.



About Mindtree

Mindtree [NSE: MINDTREE] is a global technology consulting and services company, helping enterprises marry scale with agility to achieve competitive advantage. "Born digital," in 1999 and now a Larsen & Toubro Group Company, Mindtree applies its deep domain knowledge to 260 enterprise client engagements to break down silos, make sense of digital complexity and bring new initiatives to market faster. We enable IT to move at the speed of business, leveraging emerging technologies and the efficiencies of Continuous Delivery to spur business innovation. Operating in 24 countries across the world, we're consistently regarded as one of the best places to work, embodied every day by our winning culture made up of over 27,000 entrepreneurial, collaborative and dedicated "Mindtree Minds."