PyTorch is an open-source deep learning framework that's known for its flexibility and ease-of-use. This is enabled in part by its compatibility with the popular Python high-level programming language favored by machine learning developers and data scientists.

PyTorch is a fully featured framework for building deep learning models, which is a type of machine learning that's commonly used in applications like image recognition and language processing. Written in Python, it's relatively easy for most machine learning developers to learn and use. PyTorch is distinctive for its excellent support for GPUs and its use of reverse-mode auto-differentiation, which enables computation graphs to be modified on the fly. This makes it a popular choice for fast experimentation and prototyping.

PyTorch is the work of developers at Facebook AI Research and several other labs. The framework combines the efficient and flexible GPU-accelerated backend libraries from Torch with an intuitive Python frontend that focuses on rapid prototyping, readable code, and support for the widest possible variety of deep learning models. Pytorch lets developers use the familiar imperative programming approach, but still output to graphs. It was released to open source in 2017, and its Python roots have made it a favorite with machine learning developers.

Significantly, PyTorch adopted a Chainer innovation called reverse-mode automatic differentiation. Essentially, it's like a tape recorder that records completed operations and then replays backward to compute gradients. This makes PyTorch relatively simple to debug and well-adapted to certain applications such as dynamic neural networks. It's popular for prototyping because every iteration can be different.

PyTorch is especially popular with Python developers because it's written in Python and uses that language's imperative, define-by-run eager execution mode in which operations are executed as they are called from Python. As the popularity of the Python programming language persists, a survey identified a growing focus on AI and machine learning tasks and, with them, greater adoption of related PyTorch. This makes PyTorch a good choice for Python developers who are new to deep learning, and a growing library of deep learning courses are based on PyTorch. The API has remained consistent from early releases, meaning that the code is relatively easy for experienced Python developers to understand.

PyTorch's particular strength is in rapid prototyping and smaller projects. Its ease of use and flexibility also makes it a favorite for academic and research communities.

Facebook developers have been working hard to improve PyTorch's productive applications. Recent releases have provided enhancements like support for Google's TensorBoard visualization tool, and just-in-time compilation. It has also expanded support for ONNX (Open

Neural Network Exchange), which enables developers to match with deep learning frameworks or runtimes that work best for their applications.

There's a large and vibrant community at PyTorch.org community with excellent documentation and tutorials. The forums are active and supportive.

It's written in Python and integrated with popular Python libraries like NumPy for scientific computing, SciPy, and Cython for compiling Python to C for better performance. Because its syntax and usage are similar to Python's, PyTorch is relatively easy for Python developers to learn.

It's well supported by major cloud platforms.

The scripting language, called TorchScript, is easy to use and flexible when in eager mode. This isa quick-start execution mode in which operations are executed immediately as they're called from Python, but which can also be transitioned to graph model for speed and optimization in C++ runtime environments.

It supports CPU, GPU, and parallel processing, as well as distributed training. This meana that computational work can be distributed among multiple CPU and GPU cores, and training can be done on multiple GPUs on multiple machines.

PyTorch supports dynamic computational graphs, enabling network behavior to be changed at runtime. This provides a major flexibility advantage over the majority of machine learning frameworks, which require neural networks to be defined as static objects before runtime.

The PyTorch Hub is a repository of pre-trained models that can be invoked, in some cases with just a single line of code.

New custom components can be created as subclasses of a standard Python class, parameters can easily be shared with external toolkits like TensorBoard, and libraries can easily be imported and used inline.

PyTorch has a well-regarded set of APIs that can be used to extend core functionality.

It supports both "eager mode" for experimentation and "graph mode" for high-performance execution.

It has a large collection of tools and libraries in areas ranging from computer vision to reinforcement learning.

It supports a pure C++ front-end interface that is familiar to Python programmers and can be used to build high-performance C++ applications.

PyTorch and TensorFlow are similar in that the core components of both are tensors and graphs.

Tensors are a core PyTorch data type, similar to a multidimensional array, used to store and manipulate the inputs and outputs of a model, as well as the model's parameters. Tensors are similar to NumPy's ndarrays, except that tensors can run on GPUs to accelerate computing.

Neural networks transform input data by applying a collection of nested functions to input parameters. The goal of deep learning is to optimize these parameters (consisting of weights and biases, which in PyTorch are stored in tensors) by computing their partial derivatives (gradients) with respect to a loss metric. In forward propagation, the neural network takes the input parameters and outputs a confidence score to the nodes in the next layer until the output layer is reached where the error of the score is calculated. With backpropagation inside a process called gradient descent, the errors are sent back through the network again and the weights are adjusted, improving the model.

Graphs are data structures consisting of connected nodes (called vertices) and edges. Every modern framework for deep learning is based on the concept of graphs, where Neural Networks are represented as a graph structure of computations. PyTorch keeps a record of tensors and executed operations in a directed acyclic graph (DAG) consisting of Function objects. In this DAG, leaves are the input tensors, roots are the output tensors.

In many popular frameworks, including TensorFlow, the computation graph is a static object. PyTorch is based on dynamic computation graphs, where the computation graph is built and rebuilt at runtime, with the same code that performs the computations for the forward pass also creating the data structure needed for backpropagation. PyTorch is the first define-by-run deep learning framework that matches the capabilities and performance of static graph frameworks like TensorFlow, making it a good fit for everything from standard convolutional networks to recurrent neural networks.

The PyTorch framework is known to be convenient and flexible, with examples covering reinforcement learning, image classification, and natural language processing as the more common use cases.

From Siri to Google Translate, deep neural networks have enabled breakthroughs in machine understanding of natural language. Most of these models treat language as a flat sequence of words or characters and use a kind of model called a recurrent neural network (RNN) to process this sequence. But many linguists think that language is best understood as a hierarchical tree of phrases, so a significant amount of research has gone into deep learning models known as recursive neural networks that take this structure into account. While these models are notoriously hard to implement and inefficient to run, PyTorch makes these and other complex natural language processing models a lot easier. Salesforce is using PyTorch for NLP and multi-task learning.

PyTorch is a popular choice for research due to its ease of use, flexibility and rapid prototyping. Stanford University is using PyTorch's flexibility to efficiently research new algorithmic approaches.

Udacity is educating AI innovators using PyTorch.

PyTorch is relatively simple to learn for programmers who are familiar with Python. It offers easy debugging, simple APIs, and compatibility with a wide range of extensions built-in Python. Its dynamic execution model is also excellent for prototyping, although it extracts some performance overhead.

PyTorch supports a variety of features that make the deployment of the AI models quick and easy. It also has a rich ecosystem of libraries like Captum (for model interpretability), skorch (scikit-learn compatibility), etc. to support the development. PyTorch has a great ecosystem for accelerators like Glow for Training and NVIDIA®