**FASTAI**

-A.DEEKSHANA

**ABSTRACT:**

“Now, the world is too faster. And so be a excellent code creator!!!”

Let’s go thought it!

[Fastai](http://dev.fast.ai/) is a deep learning library which provides practitioners with high-level components that can quickly and easily provide state-of-the-art results in standard deep learning domains, and provides researchers with low-level components that can be mixed and matched to build new approaches. It aims to do both things without substantial compromises in ease of use, flexibility, or performance.

**CONTENT:**

General Introduction

History of fastai

High level API foundation

Mid level APIs

Low-level APIs

Application

Control resource

NPL of text concept

Fastai.2

Conclusion

Reference link

Github link

**GENERAL INTRODUCTION:**

Fastai is organized around two main design goals: to be approachable and rapidly productive, while also being deeply hack able and configurable. Other libraries have tended to force a choice between conciseness and speed of development, or flexibility and expressivity, but not both. We wanted to get the clarity and development speed of Keras and the customizability of PyTorch.

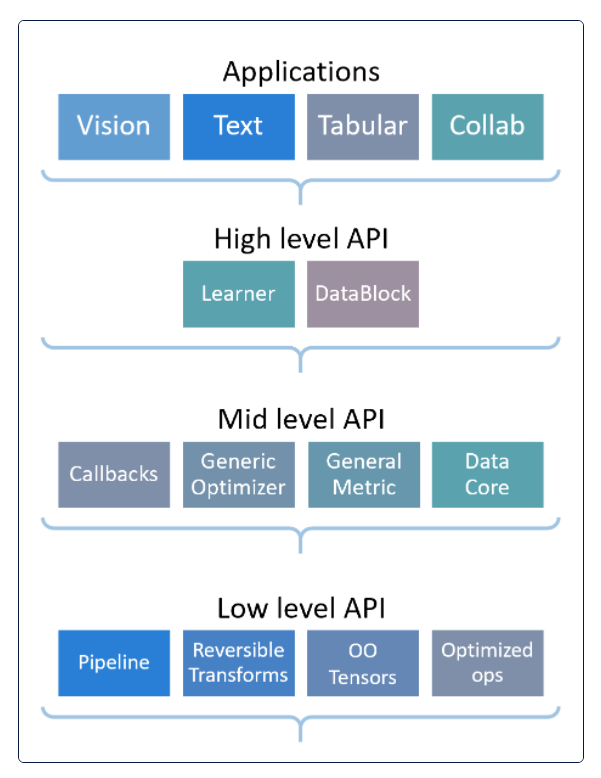
**HISTORY OF FASTAI:**

It was founded in 2016 by [Jeremy Howard](https://en.wikipedia.org/wiki/Jeremy_Howard_(entrepreneur)) and [Rachel Thomas](https://en.wikipedia.org/wiki/Rachel_Thomas_(academic)) with the goal of democratising deep learning. They do this by providing a [massive open online course](https://en.wikipedia.org/wiki/Massive_open_online_course) (MOOC) named "Practical Deep Learning for Coders," which has no other prerequisites except for knowledge of the programming language.

In the fall of 2018, fast.ai released v1.0 of their free open source library for deep learning called **fastai** . The Fast.ai algorithm was trained on the ImageNet database in 18 minutes using 16 Amazon Web Service instances, at a total compute cost of around $40. Howard claims this is about 40 percent better than Google’s effort, although he admits comparison is tricky because the hardware is different.

### HIGH-LEVEL API FOUNDATION:

A high-level API powers ready-to-use functions to train models in various applications, offering customizable models with sensible defaults. It is built on top of a hierarchy of lower level APIs which provide compostable building blocks. This way, a user wanting to rewrite part of the high-level API or add particular behaviour to suit their needs doesn’t have to learn how to use the lowest level.

****

## **MID-LEVEL APIs:**

The mid-level API provides the core deep learning and data-processing methods for each of these applications, and low-level APIs provide a library of optimized primitives and functional and object-oriented foundations, which allows the mid-level to be developed and customised.

Many libraries, including fastai version 1 or earlier, provide a high-level API to users, and a low-level API used internally for that functionality, but nothing in between. This has two problems: the first is that it becomes harder and harder to create additional high-level functionality, as the system becomes more sophisticated, because the low-level API becomes increasingly complicated and cluttered. The second problem is that for users of the system who want to customize and adapt it, they often have to rewrite significant parts of the high-level API, and understand the large surface area of the low-level API in order to do so. This tends to mean that only a small dedicated community of specialists can really customize the software.

## **LOW-LEVEL APIs**:

The middle layer is programmed in that set of abstractions. The low-level of the fastai stack provides a set of abstractions for:

* Pipelines of transforms: Partially reversible composed functions mapped and dispatched over elements of tuples
* Type-dispatch based on the needs of data processing pipelines
* Attaching semantics to tensor objects, and ensuring that these semantics are maintained throughout a Pipeline
* GPU-optimized computer vision operations
* Convenience functionality, such as a decorator to make patching existing objects easier, and a general collection class with a NumPy-like API.

The rest of this section will explain how the transform pipeline system is built on top of the foundations provided by PyTorch, type dispatch, and semantic tensors, providing the flexible infrastructure needed for the rest of fastai.

**Application:**

### DEVELOPMENT:

fastai is mostly focused on model training, but once this is done you can easily export the PyTorch model to serve it in production. The command Learner.export will serialize the model as well as the input pipeline (just the transforms, not the training data) to be able to apply the same to new data.

The library provides Learner.predict and Learner.get\_preds to evaluate the model on on item or a new inference DataLoader. Such a DataLoader can easily be built from a set of items with the command test\_dl.

## **Controlling Resources**

You would need to use an Nvidia GPU. Unless you own a gaming PC, you’re unlikely to have an Nvidia GPU. Even if you’ve got an Nvidia GPU, you may need to download any data to train models from the internet. The best option is to use a GPU in the Cloud.

Here are a few options:

### CRESTLE

Crestle is the easiest starter option. It costs about 0.60 USD an hour. You don’t have to set up anything, only start learning.

### PAPER SPACE

Paperspace has an array of GPU options to choose from. The starting option is cheaper and more reliable than Crestle. The downside is you’d have to spend around an hour doing the initial setup. If you don’t have some Linux experience, it could be not easy.

### AWS AND GCE

Both of these cloud providers have a wide range of GPU options, and in the long run, they might be more cost-effective than Paperspace. You could also use free credits to sign up—Google for “FastAI AWS” or “FastAI GCE.”

### GOOGLE COLAB

You can get a GPU for free using Google CoLab. Running fast.ai on CoLab is challenging, as it is mainly built for Tensorflow (Google’s deep learning toolkit). If you have time to experiment with that, you might try to make it work for fast.ai.

### NLP FROM TEXT CONCEPT:

In natural language processing (NLP), perhaps the most important approach to building models is through fine tuning pre-trained language models. To train a language model in fastai requires very similar code.

from fastai.text.all import \*

path = untar\_data(URLs.IMDB\_SAMPLE)

df\_tok,count = tokenize\_df(pd.read\_csv(path/'texts.csv'), ['text'])

dls\_lm = TextDataLoaders.from\_df(df\_tok, path=path,

vocab=make\_vocab(count), text\_col='text', is\_lm=True)

learn = language\_model\_learner(dls\_lm, AWD\_LSTM, metrics=Perplexity()])

learn.fit\_one\_cycle(1, 2e-2, moms=(0.8,0.7,0.8))

Fine-tuning this model for classification requires the same basic steps:

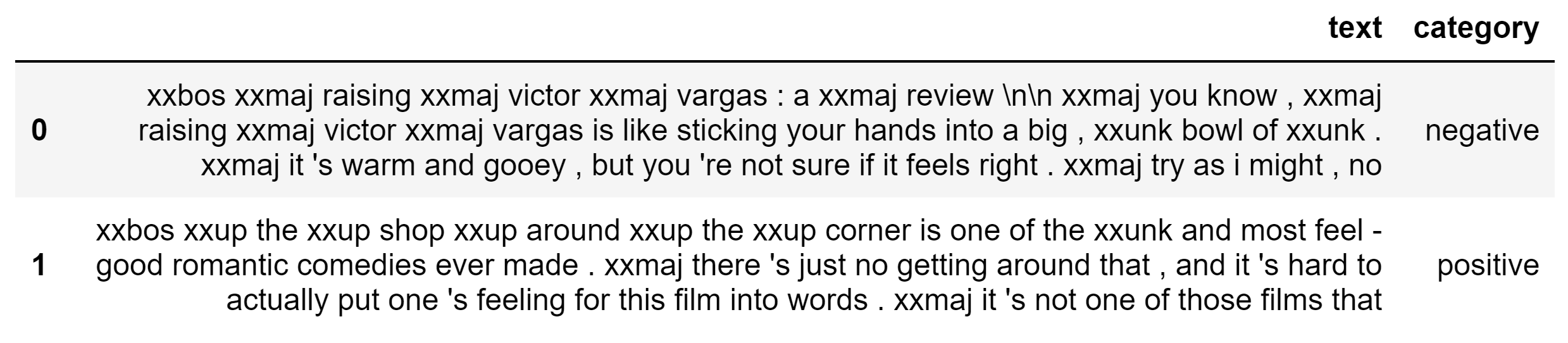
dls\_clas = TextDataLoaders.from\_df(df\_tok, path=path

vocab=make\_vocab(count), text\_col='text', label\_col='label')

learn = text\_classifier\_learner(dls\_clas, AWD\_LSTM, metrics=accuracy)

learn.fit\_one\_cycle(1, 2e-2, moms=(0.8,0.7,0.8))

The same API is also used to view the DataLoaders:



## **fastai v2:**

fastai is a deep learning library which provides practitioners with high-level components that can quickly and easily provide state-of-the-art results in standard deep learning domains, and provides researchers with low-level components that can be mixed and matched to build new approaches. It aims to do both things without substantial compromises in ease of use, flexibility, or performance. This is possible thanks to a carefully layered architecture, which expresses common underlying patterns of many deep learning and data processing techniques in terms of decoupled abstractions. These abstractions can be expressed concisely and clearly by leveraging the dynamism of the underlying Python language and the flexibility of the PyTorch library. fastai includes:

* A new type dispatch system for Python along with a semantic type hierarchy for tensors
* A GPU-optimized computer vision library which can be extended in pure Python
* An optimizer which refactors out the common functionality of modern optimizers into two basic pieces, allowing optimization algorithms to be implemented in 45 lines of code
* A novel 2-way callback system that can access any part of the data, model, or optimizer and change it at any point during training
* A new data block API

**CONCLUSION:**

Based on our experience with fastai, we believe that using a layered API in deep learning has very significant benefits for researchers, practitioners, and students. Researchers can see links across different areas more easily, rapidly combine and restructure ideas, and run experiments on top of strong baselines.

FastAI has become a popular tool for data scientists. FastAI simplifies painful aspects of model training, such as preprocessing and loading data down to a few code lines. GCP, AWS, and Azure all have already included FastAI in their machine images.

**REFERENCE LINK:**

<https://www.analyticsvidhya.com/blog/2020/10/develop-and-deploy-an-image-classifier-app-using-fastai/>

<https://www.oreilly.com/library/view/deep-learning-for/9781492045519/>

<https://www.kdnuggets.com/2019/06/things-learnt-fastai-course.html>

<https://www.bookstack.cn/read/th-fastai-book/cedc7ab42349d210.md?wd=Machine%20Learning>

<https://towardsdatascience.com/fast-ai-v2-lesson1-synopsis-tl-dr-4985bba9eea2>

<https://medium.com/featurepreneur/diving-into-fastai-with-tactlabs-920d8639e610>

<https://en.wikipedia.org/wiki/Fast.ai#:~:text=It%20was%20founded%20in%202016,of%20the%20programming%20language%20Python>.

GITHUB LINK: