HSLU Lucerne University of Applied Sciences and Arts

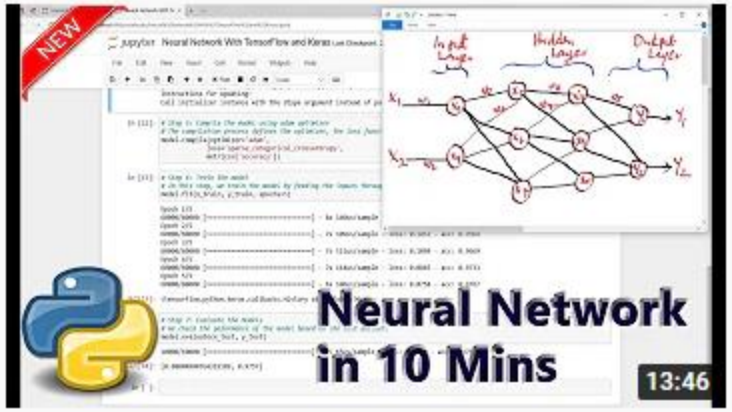
Libraries and Frameworks

NLP
Andreas Marfurt

Information Technology
21.02.2025

FH Zentralschweiz

Motivation



DATA SCIENCE AND MACHINE LEARNING

How to Build a Neural Network with TensorFlow and Keras in 10 Minutes

25K views • 2 years ago

Kindson The Genius

This is a short tutorial on How to build a Neural Network in Python with TensorFlow and **Keras** in just about **10 minutes** Full ...

CODING TUTORIALS

PyTorch Lightning from Zero to Hero (75 lines for 2022) Machine Learning/AI

3.7K views • 1 year ago

[Code] PyTorch sentiment classifier from scratch with Huggingface NLP Library (Full Tutorial)

Yannic Kilcher • 35K views • 2 years ago

Huggingface released its newest library called NLP, which gives you easy access to almost any NLP dataset and metric in one convenient interface. We will combine this with a BERT model from...

Overview

- General ML libraries
  - NumPy, SciPy
  - Scikit-learn
- NLP libraries
  - NLTK
  - Spacy
- Deep learning frameworks
  - TensorFlow
  - Pytorch
  - Keras
  - Pytorch Lightning
- Deep NLP models: Hugging Face 

General ML Libraries

- NumPy: Numerical processing, basis for many other libraries
- SciPy: Basic matrix processing algorithms
- scikit-learn: Simple and efficient ML algorithms, best for small data

All open source!

NumPy

- NumPy = Numerical Python
- Basic building block: ndarray
  - n-dimensional array object (= tensor)
    - 0D: scalar
    - 1D: vector
    - 2D: matrix
- Library with a huge amount of transformations and computations with ndarrays

NumPy ndarray

```
>>> import numpy as np
>>> vector = np.array([1, 2, 3, 4, 5])
>>> vector
array([1, 2, 3, 4, 5])

>>> vector.shape
(5,)
```

NumPy ndarray

```
>>> matrix = np.array([[1, 2], [3, 4], [5, 6]])
>>> matrix
array([[1, 2],
       [3, 4],
       [5, 6]])

>>> matrix.ndim
2
```

SciPy

- SciPy = Scientific Python
- Mathematical algorithms and helper functions built on top of NumPy
  - Linear algebra
  - Optimization
  - Information theory
  - Sparse matrices
  - ...

scikit-learn

- Machine learning algorithms built on NumPy and SciPy
  - Classification
  - Regression
  - Clustering
  - Dimensionality reduction
  - Model selection
  - Preprocessing

NLP Libraries

NLP Libraries

- NLTK: Preprocessing (tokenization, stemming, stopword removal, …)
- spaCy: NLP pipelines
- Gensim: Topic modeling

NLTK

- NLTK = Natural Language Toolkit
- A lot of NLP algorithms for preprocessing, modeling, evaluation
- NLP corpora: Gutenberg books, Reuters, stopword lists for 11 languages, WordNet (word hierarchies, synonyms)
- We're mostly going to use it for preprocessing
  - Tokenization
  - Stopword removal

spaCy

- 72+ languages
- Pretrained NLP pipelines including models
  - Part-of-speech tagging
  - Named entity recognition
  - Lemmatization
  - Dependency parsing
  - Entity linking
  - ...

spaCy

Edit the code & try spaCy spaCy v3.4 · Python 3 · via Binder

```
# pip install -U spacy
# python -m spacy download en_core_web_sm
import spacy

# Load English tokenizer, tagger, parser and NER
nlp = spacy.load("en_core_web_sm")

# Process whole documents
text = ("When Sebastian Thrun started working on self-driving cars at "
        "Google in 2007, few people outside of the company took him "
        "seriously. “I can tell you very senior CEOs of major American "
        "car companies would shake my hand and turn away because I wasn’t "
        "worth talking to,” said Thrun, in an interview with Recode earlier "
        "this week.")
doc = nlp(text)

# Analyze syntax
print("Noun phrases:", [chunk.text for chunk in doc.noun_chunks])
print("Verbs:", [token.lemma_ for token in doc if token.pos_ == "VERB"])

# Find named entities, phrases and concepts
for entity in doc.ents:
    print(entity.text, entity.label_)
```

RUN

Deep Learning Frameworks

Deep Learning Frameworks

- Model definition
  - TensorFlow: Google's deep learning framework
  - Keras: Simple model definition for common use cases, built on TensorFlow 2.0
  - PyTorch: Open-source DL framework, used in research, large ecosystem of libraries
- Training
  - PyTorch Lightning: Open-source library that standardizes training, abstracts away details such as loading to CPU/GPU, batch size, gradient accumulation, model checkpointing, early stopping, ...

Article: Which DL framework should you use?

TensorFlow

- Open-source
- Python and C++ (efficiency)
- Current version: 2
  - Earlier versions had a static computation graph: No loops, less flexibility
  - Switch to dynamic computation graph (to match competitor PyTorch)
- More performance: JAX
  - Compiles code for optimal use on GPU/TPUs
  - More flexibility: FLAX (integrated in Hugging Face transformers)
  - We're not going to use JAX in this course, but feel free to try it out!

Keras

- Simple model definition for TensorFlow
- Pretrained models
- Dataset loaders

```
tf.keras.applications.ResNet50(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    **kwargs
)
```

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

PyTorch

- Flexible, heavily used in research and prototyping
- Early days neural network library: Torch
  - Model definitions in Lua
  - Don't worry, we won't touch that
- Large Python ecosystem of libraries and extensions

PyTorch Lightning

- Out of NYU and Facebook AI Research
- Adds structure to model definition and training
- Removes boilerplate code
- Allows changing all under-the-hood functionality by overriding the respective functions
- I’m using this for all my NN training

PyTorch Lightning

- Basic unit: LightningModule
- Define the following functions
  - __init__: model definition
  - forward
  - configure_optimizers
  - training_step
  - validation_step
- Trainer object controls all training details
  - Highly configurable

Exercise: PyTorch

Hugging Face 

Hugging Face: Pretrained NLP Models

- De-facto standard for open NLP models
- All libraries are open-source
- Started by reimplementing research papers
- Large model hub
- One-stop-shop: Includes tokenizers, training helpers, datasets, accelerators (to train on GPU/TPU), evaluation metrics

Hugging Face Transformers

- Library for (almost) all Transformer models
- Well-tested and maintained by the community
- Supports PyTorch, TensorFlow and JAX
- Loading and running a pretrained model is as easy as:

```
from transformers import BertTokenizer, BertModel
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained("bert-base-uncased")
text = "Replace me by any text you'd like."
encoded_input = tokenizer(text, return_tensors='pt')
output = model(**encoded_input)
```

Hugging Face Models

- Model hub at: https://huggingface.co/models
- Browsable collection of all uploaded models that run with Hugging Face code
- Freely available versions of many major pretrained models (BERT, BART, T5, GPT-2, …)
  - Weights of newer models like LLaMA 3.1 and Gemma 2 available after signing license agreement (check this license for commercial use!)
- "Compressed" versions of these models often also available with the prefix "distil" (distilbert, distilgpt2)
  - From a technique called *knowledge distillation*
  - Almost same model performance at much smaller size