

Deep Learning Packages and resources



Runpeng Dai

The University of North Carolina at Chapel Hill

Content

1 Introduction to PyTorch basics

2 An example on Colab environment

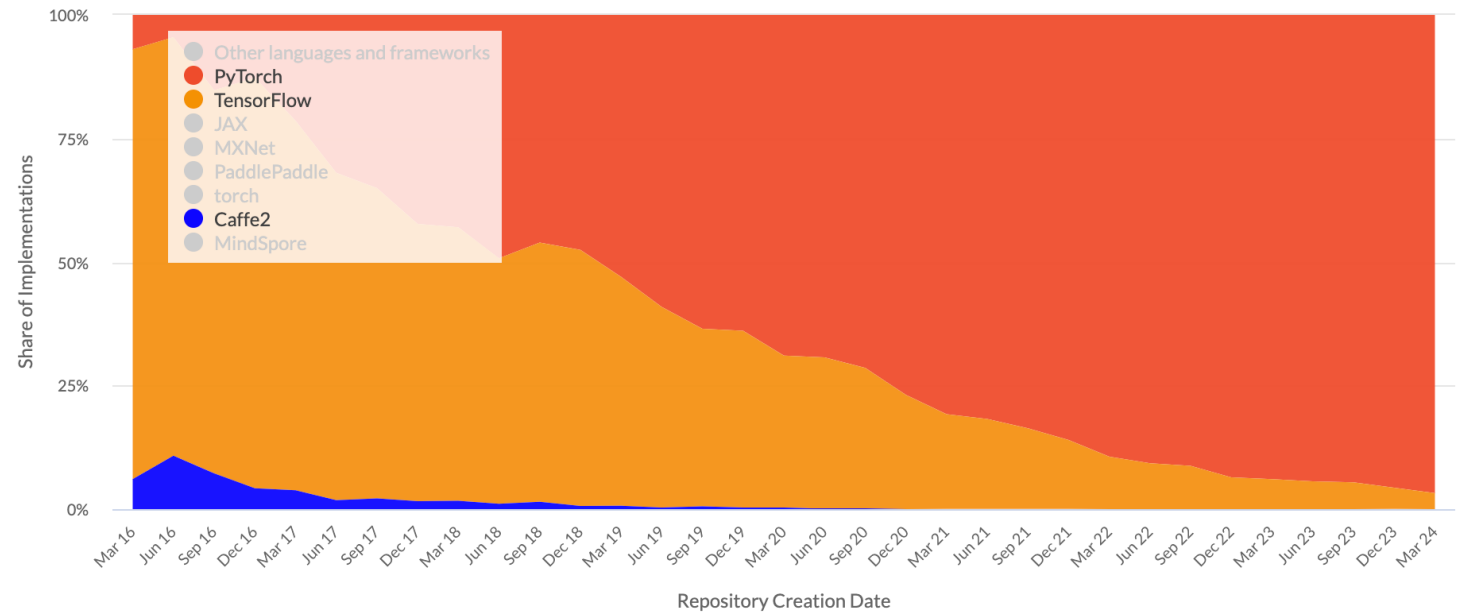
3 Resources – Hugging-face, tensorboard, W&B

Why Torch?



Frameworks

Paper Implementations grouped by framework



A simple torch code

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

# Define a simple neural network
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.dropout(x)
        x = self.fc2(x)
        return F.log_softmax(x)

# Create an instance of the network
net = Net()
```

```
# Data loading
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=32,
shuffle=True)

# Define a loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001)

# Training loop
for epoch in range(10):
    for inputs, targets in train_loader:
        optimizer.zero_grad()
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

Tensors

- Tensors are the core data structures in PyTorch, used to encode inputs and model weights.
- While many tensor operations resemble those of NumPy arrays, they are specifically optimized for deep learning and can be transferred to a GPU for accelerated processing.

Numpy style operations

```
tensor = torch.ones(4, 4)
print(f"First row: {tensor[0]}")
print(f"First column: {tensor[:, 0]}")
print(f"Last column: {tensor[..., -1]}")
tensor[:,1] = 0
print(tensor)
```

Out:

```
First row: tensor([1., 1., 1., 1.])
First column: tensor([1., 1., 1., 1.])
Last column: tensor([1., 1., 1., 1.])
tensor([[1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.]])
```

Capability of GPU computing

```
# We move our tensor to the GPU if available
if torch.cuda.is_available():
    tensor = tensor.to("cuda")
```

Easy converting from and to numpy array

```
n = np.ones(5)
t = torch.from_numpy(n)
n = t.numpy()
```

Datasets

- Torch can assist in loading and preprocessing datasets (Dataset-Dataloader pipeline). It also provides a number of pre-loaded datasets (e.g. MNIST).
- Hugging face also have similar module called Datasets.

Loading FashionMNIST

```
from torchvision import datasets
training_data = datasets.FashionMNIST(
    root="data",
    train=True,
    download=True,
    transform=ToTensor())
```

Building a Dataloader

```
from torch.utils.data import DataLoader

train_dataloader = DataLoader(training_data, batch_size=64,
                               shuffle=True)
train_features, train_labels = next(iter(train_dataloader))
```

In practice, many researches opt to write their own data processing and sampling code to gain greater flexibility.

Network structure

- In PyTorch, neural networks are defined by subclassing `nn.Module`. The network's layers and parameters are initialized within the `__init__` method, while the forward method specifies how input data flows through these layers to produce the output.
- After instantizing the network, the network does forward by passing the input data.

```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10),
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits
```

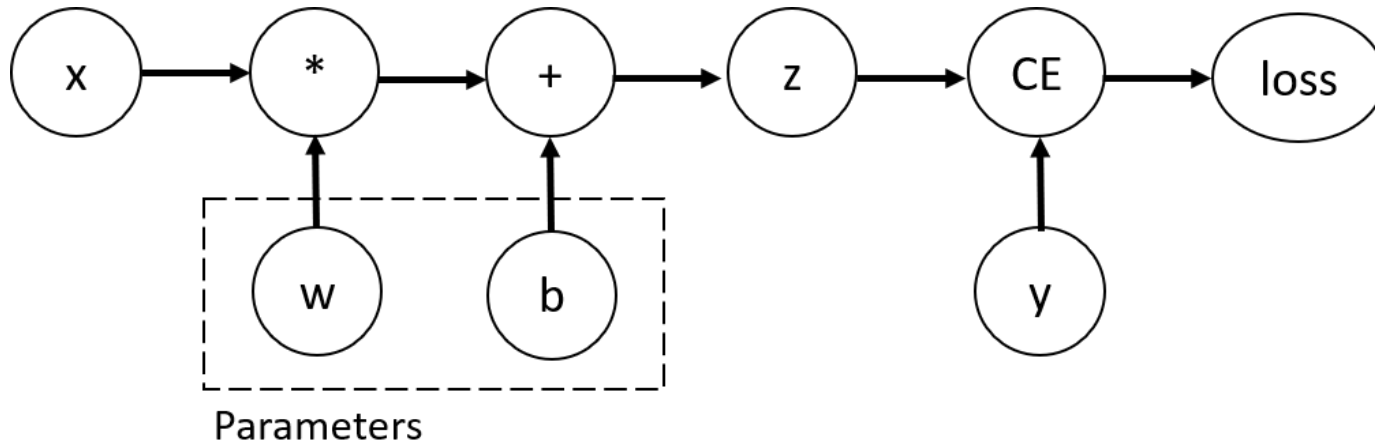
Instantize and send weights to GPU

```
model = NeuralNetwork().to(device)
X = torch.rand(1, 28, 28, device=device)
logits = model(X)
```

It is easy to implement normalizing methods (dropout batchnorm etc)

Auto differentiation

- Pytorch has a built-in engine called autograd to deal with gradients and updates.



```
x = torch.ones(5)  # input tensor
y = torch.zeros(3) # expected output
w = torch.randn(5, 3, requires_grad=True)
b = torch.randn(3, requires_grad=True)
z = torch.matmul(x, w)+b
loss = torch.nn.functional.binary_cross_entropy_with_logits(z, y)
```

```
Loss.backward()
```

- Pytorch have the notion of computation graph.
- Each time we pass input through the network (**forward pass**), PyTorch dynamically constructs a computational graph that tracks how the final output is related to all tensors requiring gradients.
- When you use backward(**backward propagation**) to calculate the grad, pytorch calculates the gradient of each component that requires_grad and store it.

Optimization

- Optimization algorithms define how model parameters adjust to reduce model error in each training step.
- Many optimization algorithms (e.g. Adam) are available in torch.optim.

```
# Define the optimizer
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)

# Within each training step, first reset the gradient of parameters
# Then calculate the gradients of the loss w.r.t. each parameter.
optimizer.zero_grad()

# Adjust the parameters using gradients
optimizer.step()
```

A simple torch code

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

# Define a simple neural network
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.dropout(x)
        x = self.fc2(x)
        return F.log_softmax(x)

# Create an instance of the network
net = Net()
```

```
# Data loading
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=32,
shuffle=True)

# Define a loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001)

# Training loop
for epoch in range(10):
    for inputs, targets in train_loader:
        optimizer.zero_grad()
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

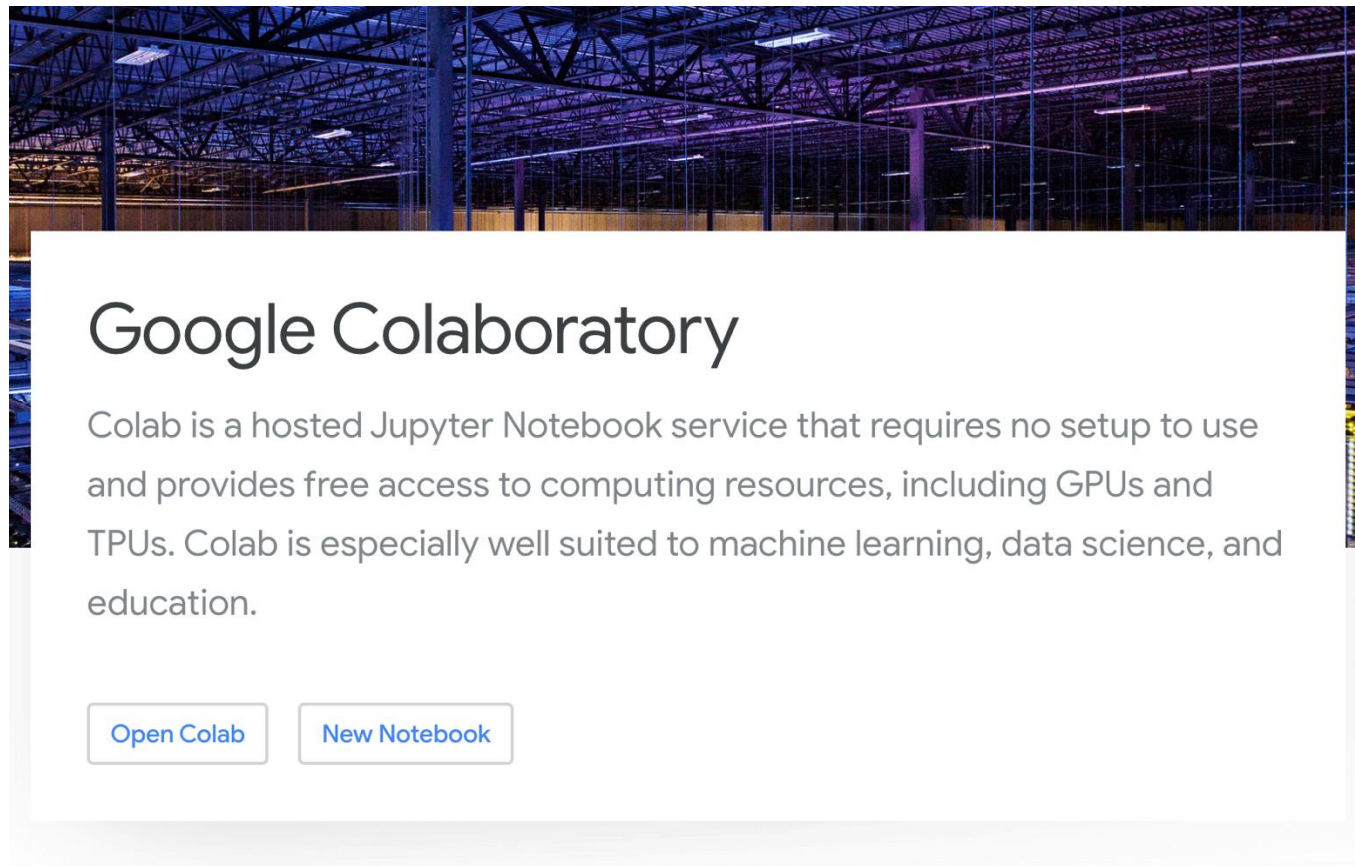
Content

1 Introduction to PyTorch basics

2 Setup the colab environment

3 Resources – Hugging-face, Paperwithcode, tensorboard, W&B

Google Colab



- Google Colab, short for Colab, is a free, cloud-based platform provided by Google Research. It allows users to write and execute Python code through a web browser.
- One of Colab's key features is the use of Jupyter Notebooks
- Colab provides free access to powerful hardware accelerators, including GPUs and TPUs
- Colab can mount google drive.

Setup

BIOS740

About

Syllabus

Staff

Calendar

Short Courses

JSM2025


ICSA2024

Q Search BIOS740

BigS2 Lab

UNC Biostatistics

Short Courses / JSM2025



Statistics, Data Science, and AI Enriching Society

Deep Learning Methods in Advanced Statistical Problems



-- JSM 2025 Short Course

Nashville, Tennessee
August 3, 2025

Introduction


This short course is designed for researchers in statistics and data analysis who are eager to explore the latest trends in deep learning and apply these methods to solve complex statistical problems. The course delves into the intersection of deep learning and statistical analysis, covering topics familiar to statisticians such as time series analysis, survival analysis, and quantile regression. Additionally, it addresses cutting-edge topics in the deep learning community, including transformers, diffusion models, and large language models. In this one-day short course participants will gain hands-on experience in exploring and applying deep learning methodologies to tackle various statistical challenges. Basic knowledge of Python programming will be helpful but not necessary.

Coding Sessions

Session	Open in Colab
Session 1: Introduction	 Open in Colab
Session 2: Generative Models	 Open in Colab

bios740


AI Mode All Images Shopping Videos News Short videos More

 GitHub

<https://bios740.github.io>

BIOS740 | Deep Learning for Biomedical Applications

May 20, 2025 — Deep Learning Methods for Biomedical Applications with PyTorch.

 The University of North Carolina at Chapel Hill

<https://www.bios.unc.edu/~dzeng/Bios740>

BIOS740 Homepage

BIOS740 covers statistical learning and personalized medicine, including topics like dynamic treatment regimes and causal inference.

- We offer two small coding sessions in the short course.
- First, open the course page <https://bios740.github.io/> and go to JSM2025
- Then open the first colab program by clicking

Code session – Pytorch Basics

- Change the runtime type to T4 GPU to have access to GPU computing resource.

The screenshot shows a Google Colab Pro environment. The top bar includes the 'packages.ipynb' title, a star icon, and a menu with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. The 'Last saved at 7:07 AM' timestamp is visible. The main code editor contains the following Python code:

```
[ ] !pip install accerlate

from transformers import AutoModelForCausalLM
device = "cuda" if torch.cuda.is_available() else "cpu"
model = AutoModelForCausalLM.from_pretrained("Qwen/Qwen1.5-0.5B")
tokenizer = AutoTokenizer.from_pretrained("Qwen/Qwen1.5-0.5B")

prompt = "Give me a short story about a cat"
messages = [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": prompt}
]
text = tokenizer.apply_chat_template(messages, tokenize=False, add_generation_prompt=True)
model_inputs = tokenizer([text], return_tensors="pt").to(device)

generated_ids = model.generate(model_inputs.input_ids, max_new_tokens=512)
generated_ids = [
```

A 'Change runtime type' dialog box is open in the center. It has a title bar 'Change runtime type'. Below the title, there is a 'Runtime type' dropdown menu set to 'Python 3'. Under 'Hardware accelerator', there are radio buttons for 'CPU', 'A100 GPU', 'L4 GPU', 'T4 GPU' (which is selected), 'TPU (deprecated)', and 'TPU v2'. At the bottom, there is a 'High-RAM' toggle switch. The dialog has 'Cancel' and 'Save' buttons.

In the top right corner, there is a 'Resources' tab. A red arrow labeled '1.' points to it. The 'Resources' tab shows 'You are subscribed to Colab Pro. Learn more', 'Available: 99.99 compute units', and 'Usage rate: approximately 0.07 per hour'. Below this, it says 'Google Compute Engine backend' and 'Resources since 7:12 AM'. A table shows 'RAM' and 'Disk' usage: '28.7 / 225.8 GB'. A red arrow labeled '2.' points to the 'Change runtime type' button at the bottom right of the dialog box.

At the bottom of the Colab interface, a status bar shows 'Connected to Python 3 Google Compute Engine backend'.

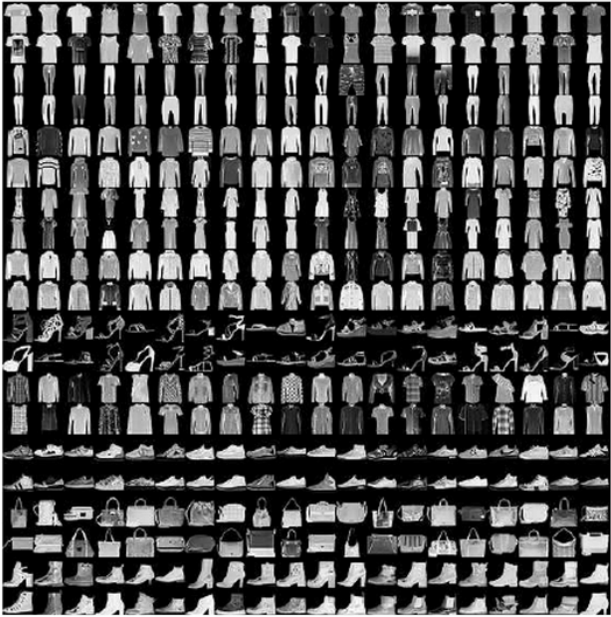
Code session – Pytorch basics

In this section, we'll work through a hands-on task together to get a practical introduction to deep learning and PyTorch coding.

FashionMNIST Image Classification

In this task, we'll classify clothing images from the FashionMNIST dataset.

We'll start with a basic fully connected neural network and then try a CNN to see how different architectures impact performance.

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Content

1 Introduction to PyTorch basics

2 Setup the colab environment

3 Resources – Hugging-face, tensorboard, W&B

Hugging Face



Hugging face is something you need to know in the **era of transformers.**

1. Transformers: A widely known python package providing state-of-the-art implementations of popular transformer based models such as VIT, BERT, GPT (Yes LLM!), and many others.
2. Model Hub: A place where users share pretrained models. Especially pretrained large language models.
3. Datasets: A library providing a wide range of datasets for different machine learning tasks.

- **Hub**

Host Git-based models, datasets and Spaces on the Hugging Face Hub.

- **Datasets**

Access and share datasets for computer vision, audio, and NLP tasks.

- **Transformers**

State-of-the-art ML for Pytorch, TensorFlow, and JAX.

Hugging Face

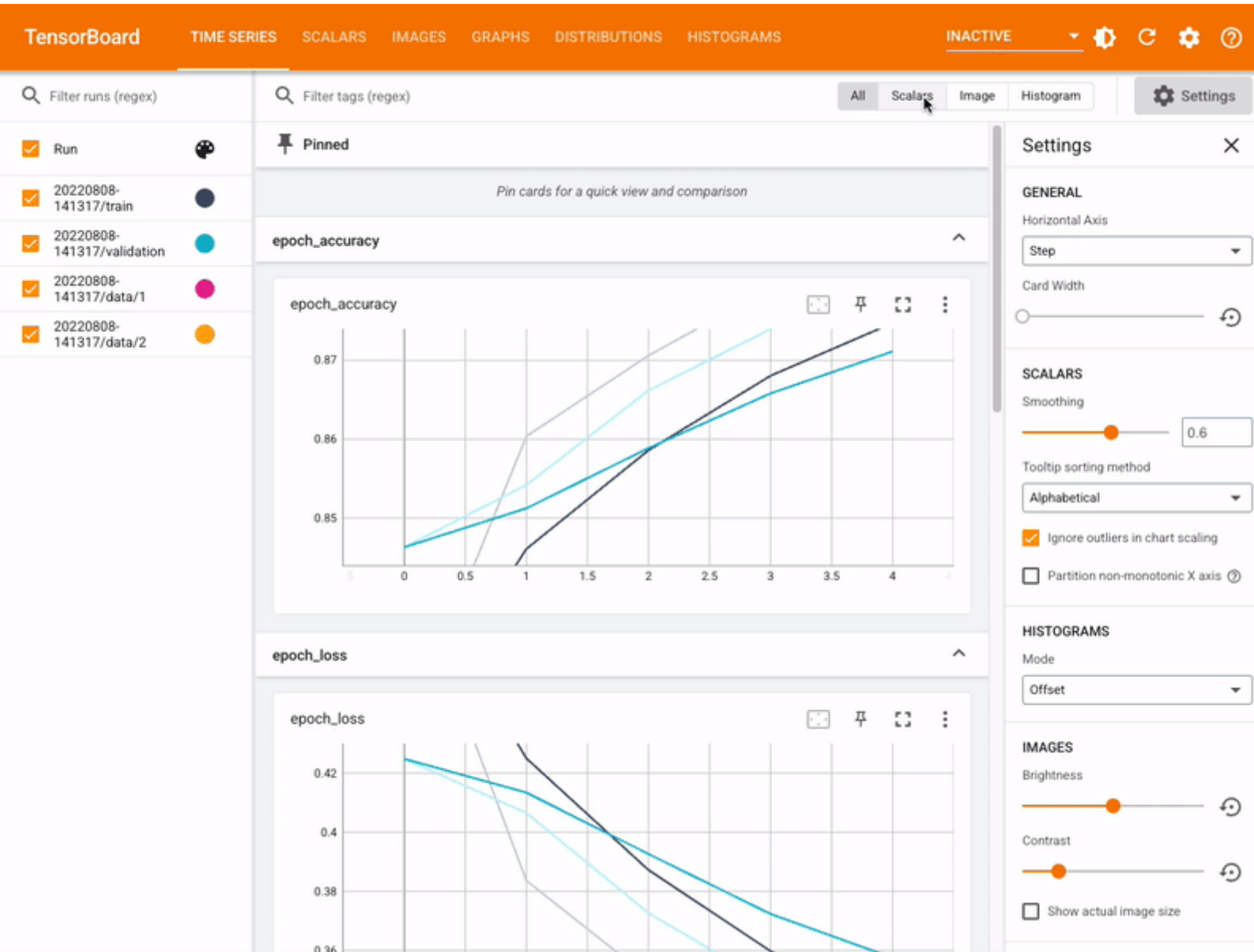
The screenshot shows the Hugging Face homepage. At the top, there's a search bar with the text "Search models, datasets, users...". Below it, there are tabs for "Tasks", "Libraries", "Datasets", "Languages", and "Licenses". A sidebar on the left lists various tasks like "Image-Text-to-Text", "Visual Question Answering", "Document Question Answering", "Depth Estimation", "Image Classification", "Object Detection", "Image Segmentation", "Text-to-Image", "Image-to-Text", "Image-to-Image", "Image-to-Video", "Unconditional Image Generation", "Video Classification", "Text-to-Video", "Zero-Shot Image Classification", "Mask Generation", "Zero-Shot Object Detection", and "Text-to-3D". The main content area displays a list of models, including "stabilityai/stable-diffusion-3-medium", "stabilityai/stable-audio-open-1.0", "2Noise/ChatTTS", "Qwen/Qwen2-72B-Instruct", "meta-llama/Meta-Llama-3-8B", "Qwen/Qwen2-7B-Instruct", and "meta-llama/Meta-Llama-3-8B-Instruct".

The screenshot shows the Hugging Face Spaces page. At the top, there's a search bar with the text "Search Spaces". Below it, there are tabs for "Browse", "ZeroGPU Spaces", "Full-text search", and "Sort: Trending". A section titled "Spaces of the week" displays a grid of AI applications, including "NPC Playground", "Phased Consistency Model P...", "Stable Audio Live Multiplayer", "GLiNER HandyLab", "Unique3D", "ToonCrafter", "CraftsMan: High-fidelity Mesh Gene...", and "Text To Image Leaderboard".

The screenshot shows the Hugging Face Datasets page for the "alpaca_eval" dataset by "tatsu-lab". The page displays the dataset card, the license (cc-by-nc-4.0), and the dataset viewer. The dataset viewer is disabled because the dataset repo requires arbitrary Python code execution, removing the loading script and relying on automated data support (you can use the datasets library). If this is not possible, please open a discussion for direct help. The README.md exists but content is empty. Use the Edit dataset card button to edit it.

Code session – Hugging face and LLM

TensorBoard



- TensorBoard is an interactive visualization tool. It is used to monitoring the training and testing process and select hyper parameters.
- It offers more flexibility than tqdm.

Weights and bias

