

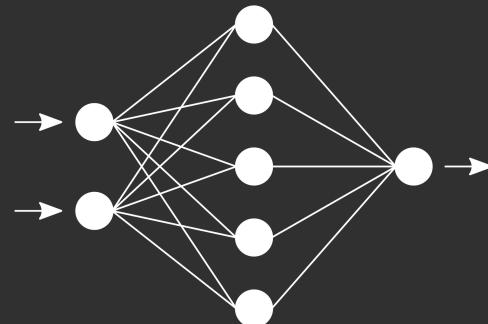


SIGMA ML

Simon Beyzerov, Aaron Tian

MACHINE LEARNING

- How can algorithms designed to make predictions improve the accuracy of their decisions without explicit instructions?
 - In real life, problems don't always fit in neat buckets
 - Perfect predictions aren't possible, but how can we get close?
 - How can we encode "learning"?
- Neural Networks are an option!

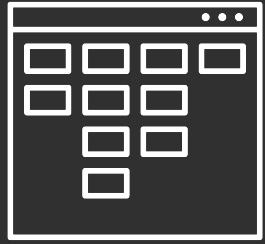




BUILDING MODELS TODAY

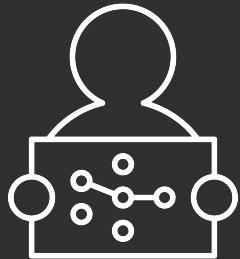
- We consider PyTorch as our primary machine learning framework
 - Future iterations may support Tensorflow
- Lack or interpretability for beginners
 - Development and experimentation with novel model architecture is difficult
 - Too much emphasis on how to implement, not how to structure the model itself
- Excessive boilerplate code
 - Boilerplate code = repetitive code that needs to be written in every program.
 - Leads to error-prone programs

GOALS



AGGREGATION

Reduce implementation complexity by organizing and combining related development steps.



EXPLAINABILITY

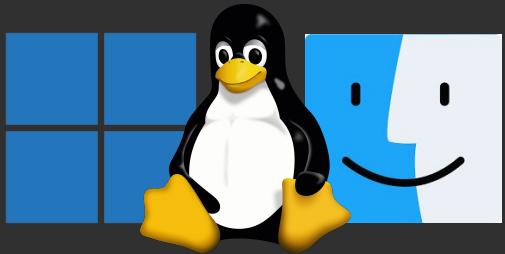
Allow for practitioners and beginners to develop a better intuition behind their models.



ANALYSIS

Encourage richer algorithmic analyses—steer emphasis away from “just make it work”.

TOOLS & APPROACHES



CROSS-PLATFORM DESKTOP APPLICATION

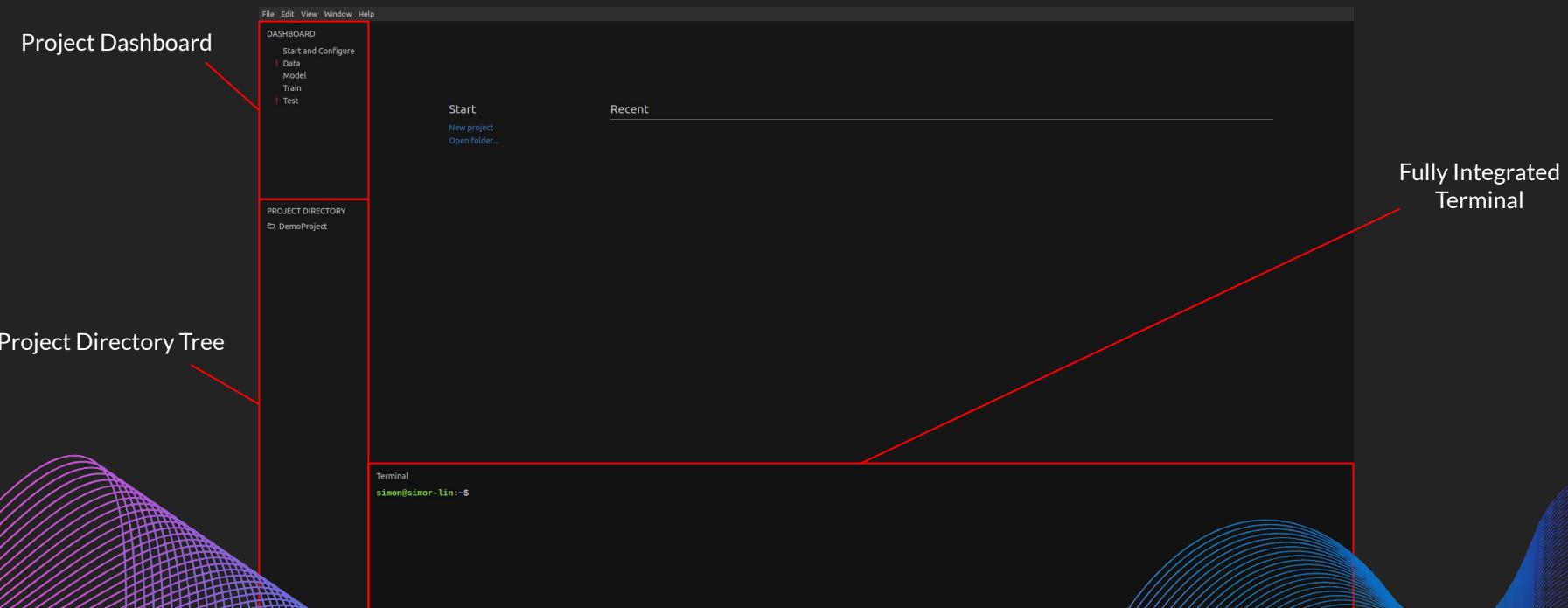
We develop a lightweight cross-platform compatible desktop application that aids in the machine-learning development process.



WEB-BASED TECHNOLOGY

Built with a flexible web-based front-end, we bundle all fundamental functionality into an intuitive UI and a modular structure that allows for future expansion.

APP & PROJECT STRUCTURE



DATA PROCESSING & DEMO

- MNIST
- Handwritten digit classification
- Data is pre-processed
- Future versions will allow for basic preprocessing

Interactive Graph-based Model

MODEL BUILDING

Adjust individual
node parameters

Drag-and-drop
Pytorch Module
Menu

The screenshot shows a dark-themed application window for building neural network models. At the top is a menu bar with File, Edit, View, Window, and Help. Below the menu is a navigation bar with DASHBOARD, PROJECT DIRECTORY (containing DemoProject), and a Start and Configure section with Data, Model, Train, and Test options.

The main workspace contains several nodes connected by dashed lines:

- A **Conv2d** node with parameters: in_channels=16, out_channels=32, kernel_size=5, stride=1, padding=2, dilation=1, groups=1, bias=True, padding_mode=zeros, device=None, dtype=None.
- An **ReLU** node with parameters: inplace=False.
- A **MaxPool2d** node with parameters: kernel_size=2, stride=None, padding=0, dilation=1.
- Another **ReLU** node with parameters: inplace=False.
- Another **MaxPool2d** node with parameters: kernel_size=2, stride=None, padding=0.

Below the workspace is a Terminal window showing the command: simon@simor-lin:~\$.

To the right of the workspace is a **MODULES** sidebar listing PyTorch modules under the **nn** category, including Conv1d, Conv2d, Conv3d, ConvTranspose1d, ConvTranspose2d, ConvTranspose3d, LazyConv1d, LazyConv2d, LazyConv3d, LazyConvTranspose1d, LazyConvTranspose2d, LazyConvTranspose3d, distance, normalization, sparse, flatten, and fold.

TRAINING

Define a search space and training parameters.

The screenshot shows the 'Training Options' section of a software interface. It includes fields for 'Epochs' (set to 7), 'Loss Function' (set to CTCloss), 'Optimization Function' (set to Adagrad, AdamW, and RAdam), and 'Learning Rate' (with sliders for 0.0001, 0.005, and 0.0002). A red box highlights this entire section.

File Edit View Window Help

DASHBOARD

Start and Configure

Data Model Train Test

PROJECT DIRECTORY

DemoProject

- compiled_model.py
- data
- model.json
- model_weights0.pt
- model_weights1.pt
- processed
- project_data.json
- raw
- temp_training_config
- training_history

Training Options

Epochs

Define the number of epochs per training iteration. The number of epochs dictates how many passes over the dataset your model performs during training.

7

Loss Function

Which loss function to use. Loss functions indicate the relative accuracy of your model's prediction during training.

CTCLoss

Optimization Function

Which optimization function to use.

Adagrad × AdamW × RAdam ×

Learning Rate

Define a search space for learning rate. Enter the minimum, maximum, and step size of the search.

0.0001

0.005

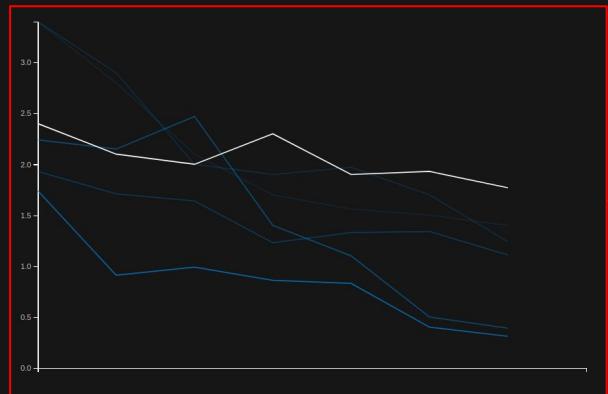
0.0002

Terminal

```
simon@simor-lin:~$
```

Top 5 hyperparameter combinations (and most recently trained)

Minimum Loss: 0.31	Minimum Loss: 0.39	Minimum Loss: 1.11	Minimum Loss: 1.24	Minimum Loss: 1.4	Minimum Loss: 1.77
--------------------	--------------------	--------------------	--------------------	-------------------	--------------------



FUTURE DIRECTIONS

- Further modularize
- More thorough data pre-processing tools
- Built-in test sets and demo models
- VCS integration
- More dynamic model building tools
- Tips and tutorials
 - + Step-by-step explanations and suggestions