

Practical Fundamentals of Deep Learning for Audio Applications

Author

Stefano Giacomelli

PhD candidate in ICT - Telecommunications Engineering, DISIM dpt. - UnivAQ

B.A. in Electronic Music Composition, M.A. in Sound Technology - MusEl dpt. ConsAQ

Contacts: stefano.giacomelli1992@gmail.com, stefano.giacomelli@graduate.univaq.it

Abstract

This workshop offers a hands-on introduction to Deep Learning for Audio Applications, specifically designed for music and sound technology students, eager to expand their expertise through both theoretical insights and practical implementations. Over several intensive sessions (3/4hrs), participants will engage in a comprehensive introductory curriculum that begins with a basic grounding in Machine Learning fundamentals and progresses toward advanced Neural Networks architectures. Early sessions focus on the essential building-blocks of Deep Learning, covering topics such as *Perceptron* models, Multi-layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs), along with specialized modules like LSTM and GRU units. As the workshop advances, participants will explore cutting-edge techniques including Autoencoders, Transformers, and Attention mechanisms, with practical exercises that enable the development and optimization of modern architectures such as Vision Transformers and Mobile Convolutional Networks. The instructional approach is both active and collaborative, ensuring that theoretical lectures are consistently integrated with hands-on coding exercises. Students will establish and exploit dual working environments—through Linux-based Virtual Machines and Google Cloud Colab—thereby gaining practical experience with nowadays essential tools and frameworks. The workshop leverages widely used Python libraries, including NumPy, PyTorch, TorchAudio, and PyTorch Lightning, to streamline the building, training, and evaluation of Deep Learning models (although no prior Python programming skill is required). This practical orientation is further enriched by guided projects that require participants to design custom modules, implement data pre-processing pipelines, and execute full train/evaluation/test workflows, all while adhering to professional software development practices, such as version control with Git. By the end of the workshop, students will acquire both the theoretical foundations and practical skills necessary to start tackling complex challenges in modern *machine listening* and audio processing research fields. Finally, available solutions for MAX/PureData integration and communication will be discussed, demonstrating how participants can effectively bridge common multimedia and audio programming frameworks with their custom-designed models, for both performative and analytical music tasks.

Schedule

Duration: 5 lectures (3/4 hours each)

Approach: active & collaborative, intermediate exercises & final project

Session(s):

- [2025a - ConsAQ \(MusEl\), Projects](#)
-

Supplemental Materials

- Reference Theory Book: <https://www.deeplearningbook.org/>
- Reference Practical Book: <https://d2l.ai/index.html>
- *Non-Reference (but useful)* Course: https://www.udemy.com/course/deeplearning_x/
(recordings available on request)
- Workshop:
 - Slides:  Slides , Colab Notebooks:  Colab Notebooks
 - Online Resources: [Neural Networks - 3Blue1Brown](#)
[Neural Networks / YouTube - StatQuest](#)
[PyTorch for DL & ML – Full Course](#)
[PyTorch Lightning Tutorials - YouTube](#)
[LightningAI - Deep Learning \(Units 1-10\)](#)
[GitHub - SopiMlab/DeepLearningWithAudio](#)
[GitHub - ACIDS/Creative Machine Learning Course](#)
 - Reference GitHub Repository:
 - Deep Torch Blocks:
<https://github.com/StefanoGiacomelli/DeepTorchBlocks>
 - Torch Audio Embeddings:
<https://github.com/StefanoGiacomelli/torch-audio-embeddings>

Workshop Lectures

1) Deep Learning Fundamentals & Basic Building Blocks

- a) Math - Machine Learning - Deep Learning Prerequisites
- b) The DL Approach: Data-driven Model Design - Training - Inference
- c) Frameworks: Python (VS-Code, G-Colab), Torch & Lightning AI, Git
- d) Dataset (1): Structures, Contents, Reading
- e) Dataset (2): Pre-Processing (MinMax VS Z-Score), Batching
- f) Dataset (3): Differentiable (& Non)-DSP, Augmentation
- g) *Perceptron*: Structure & Components
- h) Workshop Conduct & Contributing

EXERCISES

Prepare a working environment:

- *Framework 1* (Linux VM) & *Framework 2* (Google Colab)
 - Connect the environments so that can be accessible/editable from both runtimes
 - Python Packages required: `numpy`, `pytorch`, `torchaudio`, `torchvision`, `pytorch-lightning`, `torchmetrics`, `torchsummary`, `torchinfo`, `matplotlib`, `scipy`, `scikit-learn`, `pandas`
 - create a `utils.py` file and populate it with the following functions:
 - `test_runtime()`: print packages versions and available torch computing devices,
 - `gen_data()`: return an a dummy data format (`numpy.ndarray` or `torch.Tensor`) given arbitrary dimensionality specifications, `plot_data()`: generate a 1D or 2D plot of the input data, `read_file()`: read a unique or a list of audio files (from file paths) and return a `ndarray` or a `Tensor` appropriately formatted (batch=files, chs=1 or 2, samps)
 - Choose a pre-processing/augmentation pipeline and apply it to a dataset
-

2) “Basic” Architectures: MLPs - CNNs - RNNs

- a) Perceptron recap, Classification (Binary - Multi-label - Softmax) & Regression
- b) DNNs (1): Perceptrons Layering - Fully Connected Networks - MLPs
- c) DNNs (2): Convolutional Neural Networks (2D Features Processing)
- d) DNNs (3): Recurrent Neural Networks (1D Feedback Processing)
- e) Optimization (1): LSTM - GRU (1D Feedback)
- f) Optimization (2): ConvLSTM - ConvGRU (2D Feedback)

EXERCISES

Design two of the following modules (follow a standard Git-based workflow)

- Multi-Layer Perceptron → Single Output, a.k.a.: Binary classifier)
 - Multi-Layer Perceptron → Multi Output, a.k.a.: Multinomial/Softmax classifier)
 - Standard Convolutional Block
 - (Deep) Recurrent Unit
 - (Conv-)LSTM Unit
 - (Conv-) GRU
-

3) “Advanced” Architectures: AEs - Transformers

- a) DNNs (4) Auto-Encoders: Encoder, Decoder, Embedding (Latent space)
- b) DNNs (5) Transformers: Embedding
- c) DNNs (6) Transformers: Attention, Multi-Head Attention
- d) DNNs (7) Transformers: Encoder, Decoder
- e) Optimization (3): ViT, Attention algorithms, Mobile Conv-Nets

EXERCISES

Design one of the following modules (follow a standard Git-based workflow):

- Attention
- Multi-Head Attention
- ViT module
- Transformer encoder

Design one of the following architectures:

- Auto Encoder: U-Net
 - Auto Encoder: 1D/2D Conv Auto Encoder
 - Auto Encoder: De-Noising Auto Encoder
-

4) Data (Pre)-Processing & Train/Eval/Test pipelines

- a) Optimization (4) Activations, Initialization
- b) Experiment Design (1) Training: Losses, Optimizers, Epochs VS Iterations
- c) Experiment Design (2) Eval (Dev) & Test : Metrics, Backpropagation, Profiling, Reports, Plots
- d) Experiment Design (3) Integration: Checkpoints, Early Stopping, Schedulers

EXERCISES → 2 Slides for the Final Roundtable:

1. Problem and experiments formalization

2. Experiments design and structure

- Create a structured project folder and populate it with required modules and functions:
 - ./
 - |---- data/ # for audio files
 - |---- data.py # for data (pre)-processing and structuring definitions
 - |---- torch.data.Dataset
 - |---- lightning.DataModule
 - |---- model.py # for submodules and architectures definitions
 - |---- nn.Sub-Module(s)
 - |---- nn.Architecture
 - |---- pipelines.py # for train/eval/test Lightning logic definitions
 - |---- main.py # for runtime scripting (training & checkpointing)
 - |---- results/ # for result assets (textual reports, plots, checkpoints)
 - Run a train/eval/test experiment and comment results
-

5) Roundtable - HPs Search - Projects proposal

- a) HPs Search: Grid Search VS Random Search, Accelerators
- b) MAX - Pure Data integration & communication:
 - i) [NN tilde](#) (audio streaming, MAX/PD)
 - ii) shell, OSC (general purpose data streaming, cross-platform)
 - iii) [Py-Js](#) (Python interface external MAX)
- c) Final Projects:
 - i) Audio Embedding Research
 - ii) Neural Audio Building Blocks Design ([Pytorch implementation of popular Attention Mechanisms, Vision Transformers, MLP-Like models and CNNs](#))
 - iii) Individual proposals