

# 1D CNN From scratch (NumPy)

My OOP implementation of a 1D CNN (NumPy) with forward propagation and backpropagation (gradients computed by chain-rule).

**Note:** This is not a PyTorch or TensorFlow implementation — all layers, activations, loss functions, and gradient computations are manually coded for educational purpose.

## 🚀 Features

- **1D Convolution** (`Conv1D`)
- **Max & Average Pooling** (`MaxPool1D`, `AvgPool1D`)
- **Fully Connected Layers** (`FCNN`)
- **Activations:** ReLU, LeakyReLU, Sigmoid, Swish
- **Loss Functions:** Squared Error Loss (MSE), Binary Cross-Entropy (planned)
- **Forward and Backward Pass:** gradients computed by chain rule
- **Flatten Layer:** Reshapes feature maps for FCNN
- **Utility Function:** Padding, logging
- **Future additions:**
  - Batching
  - Support for multiple channels
  - Xavier/He initialization
  - Gradient clipping & numerical checks
  - Experiment tracking (W&B)
  - Vectorization
  - Tensor implementation wth CUDA support

## 📁 Repository Structure

```
.  
├── activations           #nn activations  
│   ├── LeakyReLU.py  
│   ├── ReLU.py  
│   ├── Sigmoid.py  
│   └── Swish.py  
├── config.py             #project-wide config  
└── layers                #nn layers  
    ├── AvgPool1D.py  
    ├── Conv1D.py  
    ├── FCNN.py  
    ├── Flatten.py  
    └── MaxPool1D.py  
└── loss                  #loss functions  
    ├── BCELoss.py  
    └── MSELoss.py
```

```
|   └── SquaredErrorLoss.py  
├── main.py  
├── readme.md  
├── simple_cnn.py          #1d cnn example  
└── utils  
    ├── logging_helper.py      #helper function for logging  
    ├── pad_input.py  
    └── plot_loss_curve.py
```

## ⚡ Usage Example

A working example is provided in `simple_cnn.py`

## Installation

1. Clone this repo: `git clone https://github.com/rizanB/from_scratch.git`
2. Install dependencies with pip or conda: `numpy matplotlib timeit`