NAMES: 24129097 Damien Birembaut - 24129095 Lucas Mariette

Assignment: Assignment 2 – CNN Implementation from Scratch

1. Introduction

Brief overview of the assignment goals.

> This project involved implementing a complete Convolutional Neural Network (CNN) framework from scratch in Python, without using any deep learning libraries such as PyTorch or TensorFlow. The implementation covers core CNN components, a custom training loop, and evaluation metrics, all designed to be modular and extensible.

---

2. Design and Architecture

2.1 Layer System

> Each layer (Conv2D, ReLU, Pooling, etc.) is implemented as a class with `forward()` and `backward()` methods, conforming to a base `Layer` interface for modularity.

2.2 Model Pipeline

> A custom `Model` class manages the forward and backward propagation across layers, and supports training with user-defined loss functions and optimizers.

2.3 Optimizations

- `im2col` for fast convolution
- Xavier initialization
- Support for SGD, Momentum, and Adam
- Dropout and Batch Normalization
- Regularization (L1, L2, Elastic Net)

---

3. Implemented Features

Feature	Description
Conv2D	Custom 2D convolution using im2col
ReLU & LeakyReLU	Activation layers
MaxPooling	Downsampling layer
Flatten	Transition to FC
FullyConnected	Dense layer
Dropout	Regularization
BatchNorm	Stability and faster training
Regularization	L1, L2, Elastic Net
Optimizers	SGD, Momentum, Adam
Model Save/Load	Save weights using `.npz`
Metrics	Accuracy and confusion matrix

---

4. Training and Evaluation

4.1 Dataset

Trained and evaluated on **CIFAR-10**, a 10-class image classification dataset (32×32 RGB).

4.2 Model Architecture

```
Input (3x32x32)
→ Conv2D (3→8)
→ ReLU
→ MaxPool (2x2)
→ Flatten
→ FC (8*16*16 → 10)
→ Softmax (implicitly in MSE + one-hot)
```

4.3 Training Configuration

- **Epochs**: 10
- **Batch size**: 1 (per sample forward-backward)
- **Optimizer**: SGD (momentum=0.9)
- **Regularization**: Elastic Net (λ=1e-4)
- **Loss**: MSE
- **One-hot encoded labels**

4.4 Evaluation Results

- **Test Accuracy**: `66.93%`
- **Confusion Matrix (100 batch index out of 1047)**:

```
[[785  37  50  20   3   1   9   8  43  44]
 [ 21 867   8   3   2   1   6   2  11  79]
 [ 88  14 602  75  25  29 101  28  10  28]
 [ 52  23 110 474  19  85 128  33  19  57]
 [ 54   4 126 106 385   8 193  75  16  33]
 [ 24  14  96 233  20 437  62  53  15  46]
 [  8  14  46  62   3   8 840   5   7   7]
 [ 21  13  71  76  15  31  25 702   0  46]
 [107  54  15   7   1   2  12   3 758  41]
 [ 22  86   6  10   1   1  10   4  17 843]]
```

5. Challenges & Solutions

- **Challenge**: Slow convolution
  Solution: Used `im2col` for efficient matrix-based convolution

- **Challenge**: Overfitting
  Solution: Added Dropout and L2/Elastic Net regularization

- **Challenge**: Gradient explosion/instability
  Solution: Added BatchNorm, careful initialization

6. Conclusion

The project successfully demonstrates a complete deep learning workflow implemented from scratch using only Numpy. All components from low-level layers to optimizers, metrics, and save/load are functional and tested.