"Introduction to Deep Learning"
Homework II ("practical")
version 1.0

Artem Chernodub, Ph.D.
Ukrainian Catholic University, Faculty of Applied Sciences
Grammarly
chernodub@ucu.edu.ua

July 9, 2020

1 Introduction

The homework assignments are dedicated to MLP-like neural networks with feedforward and recurrent layers. Our goal is to study their internal structure and training pipeline "under the hood".

The outcome of the practical part is a source code written in Python. You are asked to prepare the code for your equations on matrix level "from scratch". The maximum score for the Homework II is 40% of total score, submission deadline is 23/07/2020, 9:00 AM CET. The penalty for missing the deadline: up to one week – minus 50% of scores, more than one week – minus 100% of scores.

2 Tasks for Homework II

Your task is to implement training of *AutoEncNetLite* (Fig. 1) neural network from scratch. *AutoEncNetLite* is a simplified version of legendary *AutoEncNet*. The only difference here is removed recurrent connection, so all layers including $Layer_{rec}$ now are feedforward.

You got a standard *Homework II gift pack* containing 4 files:

- *train_autoenc_lite.py* - main training loop and slots for functions to be implemented;

- *utils.py* - functions and slots for functions to be implemented;
- *images_train.pickle* - training data (images);
- *images_test.pickle* - single batch of test data (images) to be visualized.

AutoEncNetLite must be trained for 1000 weights updates, batch size $N_{batch} = 20$. Training is to be done by Stochastic Gradient Descent (SGD), learning rate $\alpha = 0.001$. After training you need to run test images through the model and show inputs and outputs for them.

2.1 Initialization

1. Please, implement network's weights' initialization (see "init" method), weights $\mathbf{w}^{(in)}$, $\mathbf{w}^{(link)}$, $\mathbf{w}^{(out)}$ must be filled by Xavier uniform distribution, $\mathbf{w}^{(rec)}$ is a diagonal matrix with ones on main diagonal, biases $\mathbf{b}^{(in)}$, $\mathbf{b}^{(rec)}$, $\mathbf{b}^{(out)}$ must be filled by zeros (5% of total score).

2.2 Forward pass

1. Please, implement forward pass for $Layer_{in}$ in a scalar form. Compare it's running speed with $Layer_{in}$ in a vector form (see "forward" method) and report the results of comparison (5% of total score).
2. Please, implement forward pass for layers $Layer_{rec}$, $Layer_{link}$, $Layer_{out}$ in a vector form (add it to "forward" method) (5% of total score).

2.3 Backward pass & training

1. Please, implement sum squared loss function (see "get loss" function) (5% of total score).
2. Please, implement backpropagation (see "backprop" method). The method must return derivatives $\frac{\partial Loss}{\partial \mathbf{w}^{(in)}}$, $\frac{\partial Loss}{\partial \mathbf{b}^{(in)}}$, $\frac{\partial Loss}{\partial \mathbf{w}^{(link)}}$, $\frac{\partial Loss}{\partial \mathbf{w}^{(out)}}$, $\frac{\partial Loss}{\partial \mathbf{b}^{(out)}}$. Former recurrent layer is non-trainable now (10% of total score).
3. Use calculated derivatives to apply corrections to neural networks' weights by calling "apply_dw" method (5% of total score).
4. Plot learning curve (loss function values w.r.t. updates no.). Load test images, run it through the network and report as pairs, inputs together with outputs (5% of total score).

Good luck!

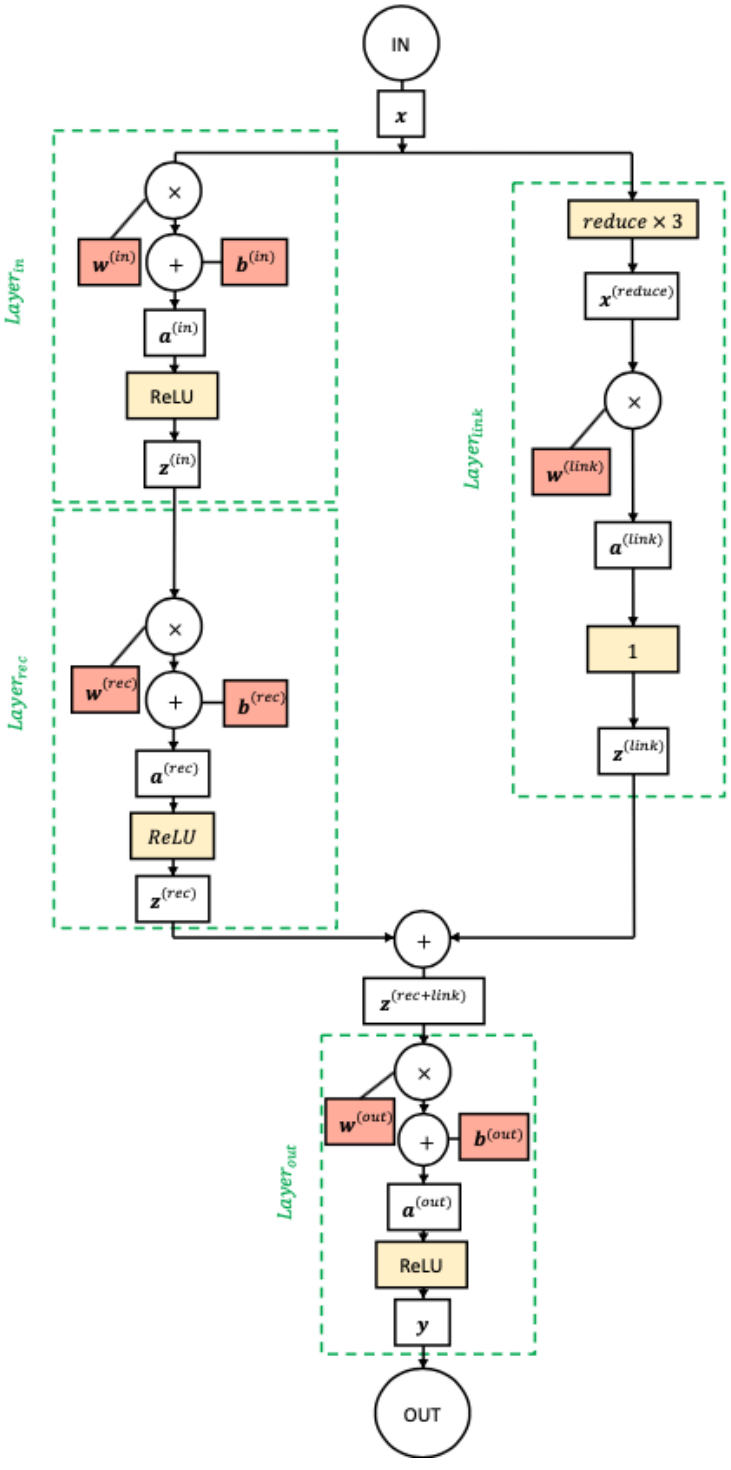

Figure 1: AutoEncNetLite's computational graph.