



**UNIVERSITÉ  
DE GENÈVE**

**FACULTÉ DES SCIENCES**

---

# DeepLearning Project 1

---

Fardin Hossain  
Salma Ennaji  
Yacine M'Rad

December 18, 2022

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Weight sharing</b>	<b>2</b>
<b>3</b>	<b>Auxiliary loss</b>	<b>2</b>
<b>4</b>	<b>Networks</b>	<b>2</b>
4.1	AlexNet . . . . .	2
4.1.1	Functions . . . . .	2
4.2	MLP . . . . .	3
<b>5</b>	<b>Results</b>	<b>3</b>
<b>6</b>	<b>Conclusion</b>	<b>5</b>

# 1 Introduction

To run the project 1: `python3 train.py`

Goal: Implements different architectures and evaluate the impact of weight sharing and auxiliary loss.

Input: Images of digits of size  $28 \times 28$  from the MNIST dataset.

Output: The network should tell us if the digit in the first image is greater or smaller than the digit in the seconde image.

## 2 Weight sharing

With weight sharing, the neural network uses the same weights while working in tandem on two different input vectors to compute comparable output vectors.

## 3 Auxiliary loss

Auxiliary loss is additional loss to help optimize the learning process of the neurones.

## 4 Networks

The main idea was to first be able to get the labels of our images and then performe a classification on the output (ie telling us if the first images is greater or smaller than the second one).

### 4.1 AlexNet

#### 4.1.1 Functions

We decided to implement an architecture based on AlexNet because it is a very powerfull and relatively simple architecture.

AlexNet is a convolutional neural network that was very famous in 2012 for winning the ILSVRC (ImageNet Large Scale Visual Recognition Challenge) competition on labeled images which aimed to achieve the best accuracy on many visual recongnition taks.

Being a CNN, AlexNet use convolutional layers which mainly correspond to a dot product multiplication between two matrix (an image and a filter).

Also AlexNet use the batch normalisation, which is a way to transform a batch of inputs and make them have a mean = 0 and a standard deviation = 1. This function aim to neutralize the effect of unstable gradient.

The maxpooling layer take a sub sample of pixel and keep as output the maximum pixel value or the average of them not sure, need to check that later.

AlexNet has also brought many improvement to the field of Deep Learning such as rectified linear units (ReLU) allowing the training part to be way faster as gradient descent optimization

is done much faster compared to other non-linearity techniques. The ReLu layers will set all the negative value of the neurone to zero and keep the positive values.

The dropout function is usfulle to avoid our model to overfit the data by reducing the dependencies between neighbouring neurons. Even tho this function has the disadvantage of making the convergence much slower.

The sigmoid activation function used for two-class classification.

Therefore we decided to use AlexNet for solving our image clasification's problem.

## 4.2 MLP

We implement a multilayer peceptron neural network to perform a comparaison task.

We implemented an MLP to compare the output of our classifier. Inded we used AlexNet as a classifier ie to give labels to our input images, then the idea was to use the output of our classifier which is a tensor of 20 elements with 10 classes for each input. Knowing the label we used MLP to classify our MNIST images.

## 5 Results

We have implemented 3 architectures :

- An AlexNet CNN to tells us which images is greater than the other ie compare them. (No labels
- An AlexNet CNN to tells us the labels of our input images + an MLP NN to compare the images using weight sharing.
- An AlexNet CNN to tells us the labels of our input images + an MLP NN to compare the image using weights sharing and the auxiliary loss.

Hyper-parameters :

- Learning rate : 0.001
- Batch size : 100
- Epochs : 30

Here are the obtained results on  $N = 1000$  samples for 10 runs:

Architecture	mean	std dev
AlexNet alone	98.16	0.39
Weight Sharing + AlexNet + MLP	97.52	0.53
Weight Sharing + Auxiliary loss + AlexNet + MLP	94.25	0.59

Table 1: Statistics on **the train** for the accuracy

Architecture	mean	std dev
AlexNet alone	81.76	1.24
Weight Sharing + AlexNet + MLP	86.53	1.27
Weight Sharing + Auxiliary loss + AlexNet + MLP	87.14	1.34

Table 2: Statistics on **the test** for the accuracy

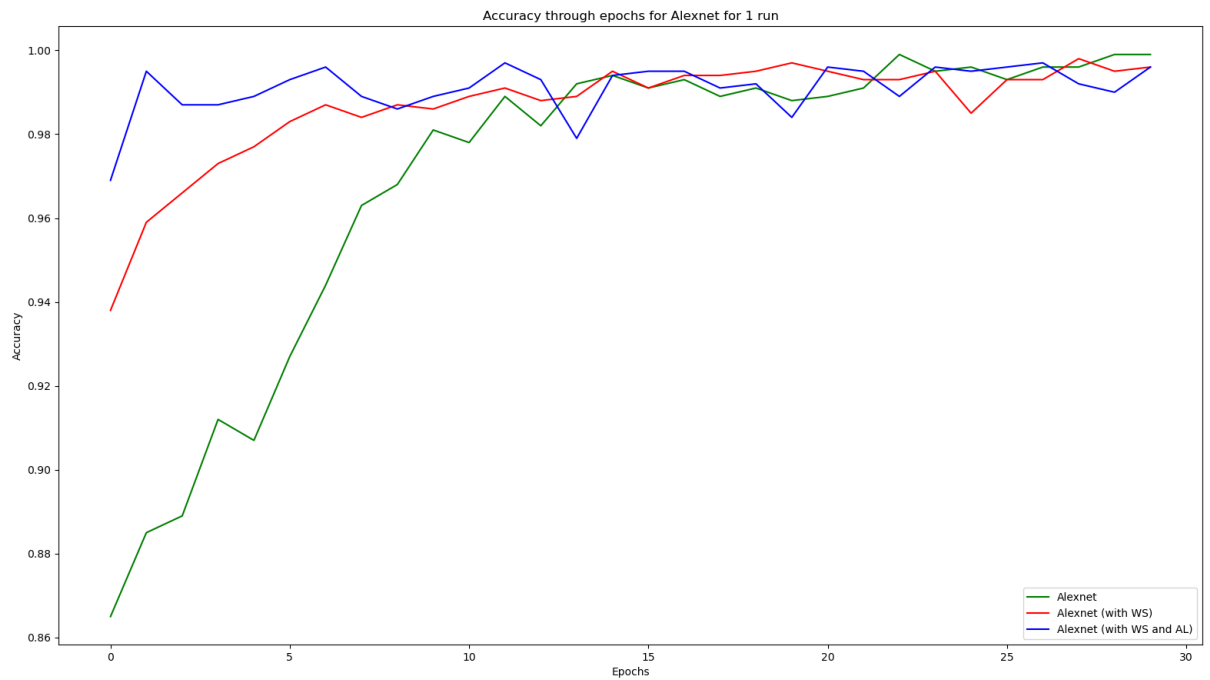


Figure 1: Evolution of the accuracy through the epochs for the 3 architectures.

## 6 Conclusion

We can see that the networks perform better when knowing the associated labels but also when combined weights sharing and auxiliary loss. We can see that our models did not overfit and were able to generalize the knowledge on the test set. Also the standard deviation remain quite small which mean our results are quite stable.

We can also see that the architecture with weights sharing and auxiliary loss converge way faster than the two others.