Deep Learning : Mini-Project 1

Massonnet Julien
Saini Anand Prakash

December 17, 2021

1 Introduction

In this part of the project, we have compared performance of different architectures aiming to compare two pictures of digits and tell if the forst one is smaller than the second.
The architectures studied in this project are Direct and Simple High Resolution (SHres) of the images.

1.1 Direct

In Direct, we are trying to solve by simply extracting features on the given images and try to make a classification on the images.

1.2 SHres

In Simple High resolution (SHres), we have used Upsampling methods and increased the spatial resolution of the images from 14*14 to 28*28 using the nearest-neighbor interpolation algorithm. Which selects the value of the nearest pixel for each position to be interpolated regardless of any other pixel. Then applied smoothing and sharpening the edge before the classification.

1.3 complementary techniqus

On both the architectures, we studied the impact of weight sharing (WS), using Auxiliary loss (AL) and Asymmetry (ASYM).

1.3.1 Weight sharing

Weight sharing in convolutional Neural Networks (CNN) is a technique to reuse the same weights on nodes reducing the total number of weights in a training network. CNN works by implementing filters of certain strides over the input image. For example, an image of 14*14 has a filter of 2*2 with a stride of size 2. The filter used is of four weights (one per pixel), it is applied

four times and making it 16 eights in total. With weight sharing we use the same weights across all the four filters.

1.3.2 Auxiliary loss

Auxiliary Losses is added in deep neural network in order to tackle the vanishing gradient problem. The aim of using these is to improve the convergence of deep networks during training by pushing gradients to the lower layers.

1.4 Asymmetry

We know that if the input are reverse (first is second and second is first) then the ouput should be reverse also (True become False and False become True) so we're taking an auxiliary loss on the "lesser" bloc detection that correspond to reversing the input.

1.5 Parameters

The following fig show the number of multiplication by a parameters each models haves :

Covnet	Parameters
Direct_basic	2(1*6*3*3) + 2(6*16) + 2(256*120) 2(120*84) + 2(840) + (20*12) + (12*8) + (8*2) = 83932
directWS	(1*6*3*3) + (6*16*3*3) + (256*120) + (120*84)+(840) + (20*12) + (12*8) + (8*2) = 42910
directAL	2(1*6*3*3) + 2(6*16*3*3) + 2(256*120) + 2(120*84)+ 2(840) + (20*12) + (12*8) + (8*2) = 85468
directALWS	42910
directALWSASYM	42910
SHresWS	(1*6*5*5) + (6*16*5*5) + (265*120) + (120*84) + (840) + (20*12) + (12*8) + (8*2) = 45622
ShresAL	2(1*6*5*5) + 2(6*16*5*5) + 2(265*120) + 2(120*84) + 2(840) + (20*12) + (12*8) + (8*2) = 90892
SHresALWS	45622
SHresALWSASYM	45622

Fig : number of parameters (for multiplication)

2 Models

Direct_basic.py : The most basics model :
- feature extractor : 2 convolution with 1 ReLU and one MaxPool
- classifier 3 Linear and 2 ReLU

direct : Rnumber: Using number4:
Feature extractor: 2 Convolution- padding, 2 ReLU and 2 Pooling Layer
Classifier: 3 Linear and 2 ReLU layer
Lesser: 3 Linear and 2 ReLU layer

SHres : Rnumber: Using number2:
Feature extractor: 2 Convolution, 2 ReLU and 2 Pooling Layer
Classifier: 3 Linear and 2 ReLU layer
Upsampling: 1 Convolution transpose
Smoothing:1 Convolution
Edge sharpening: 1 Threshold
Lesser: 3 Linear and 2 ReLU layer

3 Results

The following statistics are the results of 10 runs for each method with same optimizer (Adam) and cross entropy loss function. We looking at the error on the test data.

directALWS

- mean = 18%
- std = 2%
- best = 17%
- worst = 22%

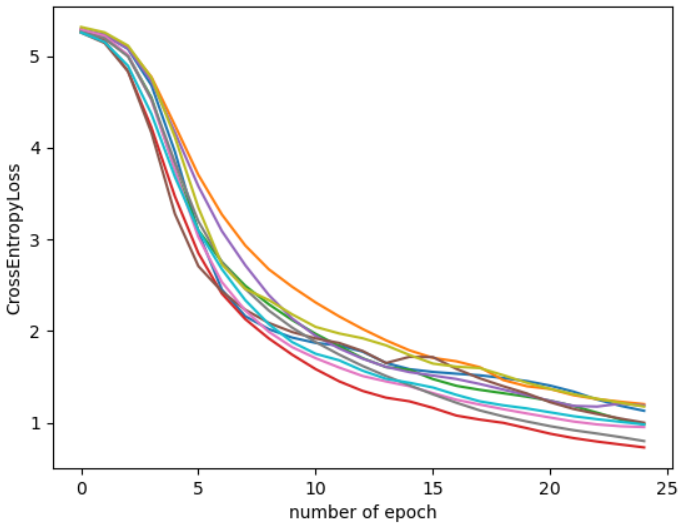

Fig : directALWS over 10 run

SHresALWS

- mean = 19%
- std = 2%
- best = 17%
- worst = 22%

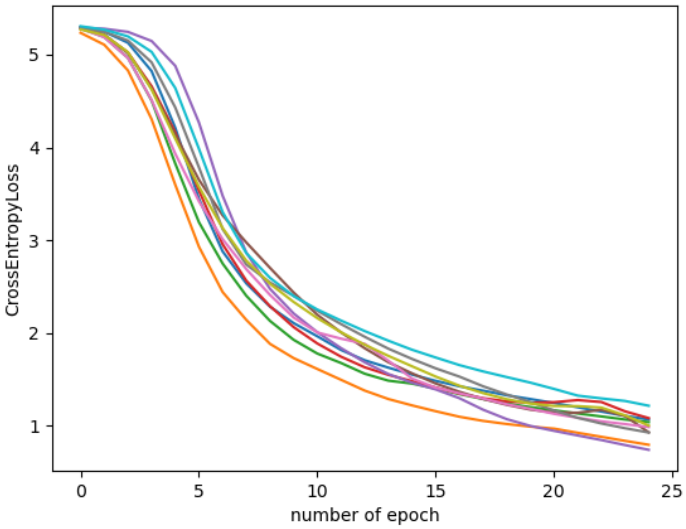

Fig : SHresALWS over 10 run

directALWSASYM

- mean = 19%
- std = 1%
- best = 17%
- worst = 20%

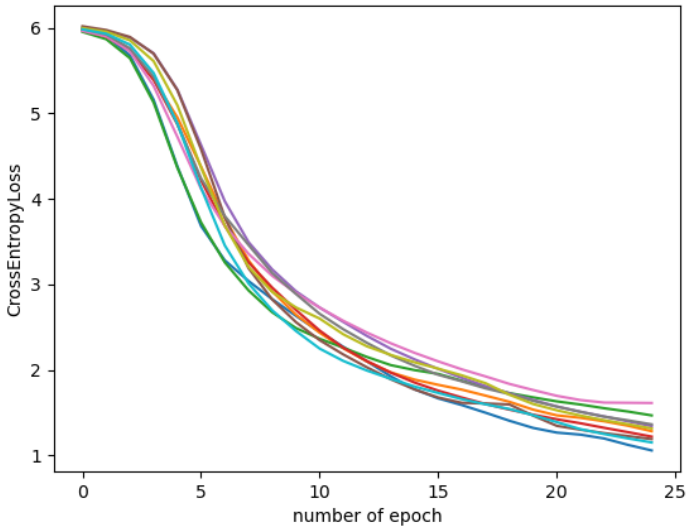

Fig : directALWSASYM over 10 run

SHresALWSASYM

long

- mean = 15%
- std = 2%
- best = 12%
- worst = 19%

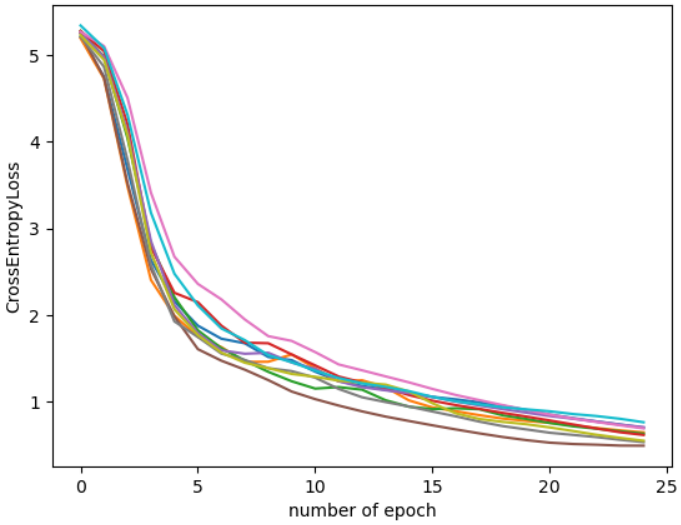

Fig : SHresALWSASYM over 10 run

directAL

- mean = 15%
- std = 2%
- best = 12%
- worst = 19%

SHresAL

- mean = 17%
- std = 2%
- best = 14%
- worst = 20%

directWS

- mean = 16%
- std = 1%
- best = 15%
- worst = 18%

SHresWS

- mean = 18%
- std = 3%
- best = 15%
- worst = 26%

Direct_basic

- mean = 21%
- std = 2%
- best = 18%
- worst = 25%

3.1 Conclusion

First we can see that having knowledge of the problem helps to improve the results, Direct_basic has the worst average error even if it has more parameters (here more parameters doesn't mean better fitting).
Auxiliary loss is better for reducing the error as compare to weight sharing, when combining both, direct has better result however in the end with every additional techniques SHres has better results with an average of 15% error and a best solution of 12%.
We have to keep in mind that SHres takes more time to train since it has more parameters.