The title of my thesis

Any short subtitle

Lucas Charpentier

Thesis submitted for the degree of
Master in Computational Science
(Imaging and Biomedical Computing)
60 credits

Departement of Informatics
Departement of Physics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2020

The title of my thesis

Any short subtitle

Lucas Charpentier

The title of my thesis

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

The title of my thesis

Lucas Charpentier

9th July 2020

Abstract

Contents

1 Introduction **1**
1.1 Background and Motivation 1
1.2 Problem Statement 1
1.3 Thesis Outline 1

2 Planning the project **3**
2.1 Machine Learning 4
2.1.1 Supervised Learning 4
2.1.2 Unsupervised Learning 4
2.2 Artificial Neural Networks 4
2.2.1 Perceptron 4
2.2.2 Multilayer Perceptron 4
2.2.3 Training a Neural Network 4
2.3 Convolutional Neural Network 4
2.3.1 Convolutional Layers 4
2.3.2 Pooling Layers 4
2.4 Neural Network Training Optimization 4
2.4.1 Weight Initialization 4
2.4.2 Training Batch Size 4
2.4.3 Dropout 4
2.5 Network Pruning 4
2.6 Datasets 4
2.6.1 MNIST 4
2.6.2 Fashion MNIST 4
2.6.3 CIFAR-10 4
2.7 Architectures 4
2.7.1 VGG-16 4

3 Single Layer ANN **5**
3.1 Pruning Nodes at Random 5
3.1.1 MNIST 5
3.1.2 Fashion MNIST 6
3.2 Estimating Node Importance based on Loss 11
3.3 Pruning network with pre-calculated importance 12
3.4 Pruning Nodes based on the Loss 12
3.5 Greedy approach to pruning instead of Exhaustive approach 12

4 Multi-Layer Perceptron **15**
4.1 Effects of Changing Training Batch Size on Node Importance 15
4.2 Effects of Using Dropout 15
4.3 Iterative weight initialization using Node importance 15

5 Convolutional Neural Network **17**
5.1 Looking at effects of per class accuracy after pruning 17
5.2 Pruning based on class accuracy 17

6 Case study: Reducing a VGG-16 model trained on X dataset **19**

7 Conclusion **21**
7.1 Summary 21
7.2 Future Works 21

List of Figures

3.1 Short title 6
3.2 Short title 8
3.3 Short title 9
3.4 Short title 9
3.5 Short title 10
3.6 Short title 10
3.7 Short title 11
3.8 Short title 12
3.9 Short title 12
3.10 Short title 13

List of Tables

3.1 Short 8
3.2 Short 8
3.3 Short 9
3.4 Short 10
3.5 Short 11
3.6 Short 11

Preface

Chapter 1

Introduction

1.1 Background and Motivation

1.2 Problem Statement

1.3 Thesis Outline

Chapter 2

Planning the project

2.1 Machine Learning

2.1.1 Supervised Learning

2.1.2 Unsupervised Learning

2.2 Artificial Neural Networks

2.2.1 Perceptron

2.2.2 Multilayer Perceptron

2.2.3 Training a Neural Network

2.3 Convolutional Neural Network

2.3.1 Convolutional Layers

2.3.2 Pooling Layers

2.4 Neural Network Training Optimization

2.4.1 Weight Initialization

2.4.2 Training Batch Size

2.4.3 Dropout

2.5 Network Pruning

2.6 Datasets

2.6.1 MNIST

2.6.2 Fashion MNIST

2.6.3 CIFAR-10

2.7 Architectures

2.7.1 VGG-16

Chapter 3

Single Layer ANN

In this chapter we will start by analyzing how effective removing nodes at random is. For this section we will only consider a single hidden layer ANN. We will then try estimating the importance of each node by using the loss function and classifying them as important, zero or worse. An important node is one that when removed would increase the loss of the model. A zero node will not significantly affect the loss of the model when removed. Lastly, a worse node will improve the loss of the model when removed. At this point we will also consider a 3 hidden layer MLP and a CNN with 4 convolutional layers followed by a dense layer in addition to the single layer ANN. By using these estimated node importance's, we still prune the models. We will start by pruning the models based on the pre-calculated node importance's, this will be slightly too extreme. Therefore, we will prune the networks by re-calculating the node importance's after removing a node till no node improves the model when removed. Initially, we will do this exhaustively by finding the node when removed will improve the model the most and then removing it. This is very time-consuming and we will therefore consider a "greedy" method that removes the first node that improves or does not affect the model, we will also start ignoring nodes that are quite important for the model in subsequent removals. For the sections on the pruning.

TODO (Describe the datasets used, and the model we consider. Talk about the what we will keep the same during the whole project and what is kept the same for this chapter)

3.1 Pruning Nodes at Random

TODO - Describe the algorithms used to remove the nodes randomly

3.1.1 MNIST

Artificial Neural Network with single hidden layer of 128 nodes, using ADAM as optimizer with a learning rate of 0.001 trained on 5 epochs, batch size of 32. Final Accuracy and Loss on test set are: Loss: 0.0879 Accuracy: 0.9724

Algorithm 1: Removing a user defined number of random nodes

```
def removeRandomNodes(n, weights, to_consider):
    Input:
    n is the number of nodes removed;
    weights are the weights of the model;
    to_consider is an array containing the nodes to consider in the
    random choosing
    Output: The weights with the nodes removed (set to zero) and
            the positions of the nodes removed
    /* Start of the code                                    */
    to_drop ← choose n from to_consider without replacement
    for i in to_drop:
        weights[0][:,i] = 0 ;      /* weights going to the node */
        weights[1][i] = 0 ;           /* bias going to the node */
        weights[2][i,:] = 0 ;      /* weights outgoing the node */
    return weights, to_drop
```

Trial text

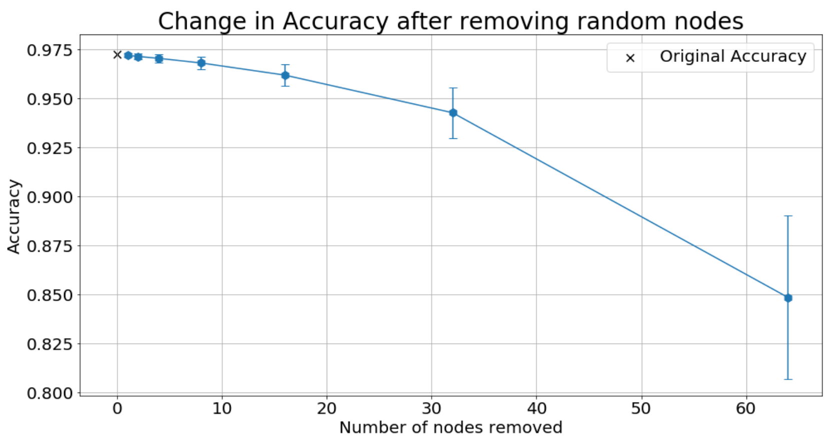

Figure 3.1: Testing

Trial text
Trial text
Trial text
Trial text
Trial text
Trial text
Trial text

3.1.2 Fashion MNIST

Trial text

Algorithm 2: Shrinking the model by removing nodes randomly

```
def shrinkModelRandomly(model, acc, loss, weights, n, to_test, x_train,
  y_train, v):
    Input:
    model is the TensorFlow model of the neural network used;
    acc is the accuracy of the original model;
    loss is the loss of the original model;
    weights are the weights of the model;
    n is the number of nodes removed at each step;
    to_test is the number of times we try to remove nodes;
    x_train is the training dataset used;
    y_train is the labels of the training dataset used;
    v defines whether are output should be verbose or not;
    Output: The weights with the nodes removed (set to zero) and
            the number of nodes removed
    /* Start of the code                                          */
    best_loss ← loss
    best_acc ← acc
    best_weights ← copy(weights)
    num_removed ← 0
    to_consider ← list from 0 to the number of nodes
    for _ in range(to_test):
        test_weights ← copy(best_weights)
        test_weights, dropped = removeRandomNodes(n, test_weights,
          to_consider)
        new_loss, new_acc ← evaluate model on x_train and y_train
        score =
          (1 − (new_loss/best_loss)) + ((new_acc/best_acc) − 1)
        if score > 0:
            best_loss ← new_loss
            best_acc ← new_acc
            best_weights ← copy(test_weights)
            increment num_removed by n
            for node in dropped:
                remove node from to_consider
    return best_weights, num_removed
```

	1	2	4	8	16	32	64
mean	0.9720	0.9714	0.9704	0.9681	0.9619	0.9427	0.8485
std	0.0009	0.0013	0.0020	0.0032	0.0056	0.0130	0.0417
min	0.9687	0.9624	0.9602	0.9523	0.9270	0.8861	0.6831
25%	0.9715	0.9708	0.9694	0.9664	0.9588	0.9364	0.8215
50%	0.9721	0.9717	0.9708	0.9685	0.9628	0.9448	0.8546
75%	0.9726	0.9724	0.9719	0.9704	0.9660	0.9526	0.8803
max	0.9736	0.9743	0.9744	0.9749	0.9724	0.9662	0.9328

Table 3.1: Long

	1	2	4	8	16	32	64
mean	0.0895	0.0914	0.0948	0.1024	0.1216	0.1794	0.4466
std	0.0026	0.0038	0.0058	0.0095	0.0164	0.0361	0.1074
min	0.0848	0.0839	0.0838	0.0844	0.0901	0.1127	0.2306
25%	0.0877	0.0887	0.0907	0.0957	0.1101	0.1516	0.3662
50%	0.0890	0.0907	0.0937	0.1012	0.1186	0.1738	0.4313
75%	0.0908	0.0932	0.0981	0.1069	0.1305	0.1980	0.5112
max	0.0997	0.1245	0.1292	0.1489	0.2233	0.3409	0.9883

Table 3.2: Long

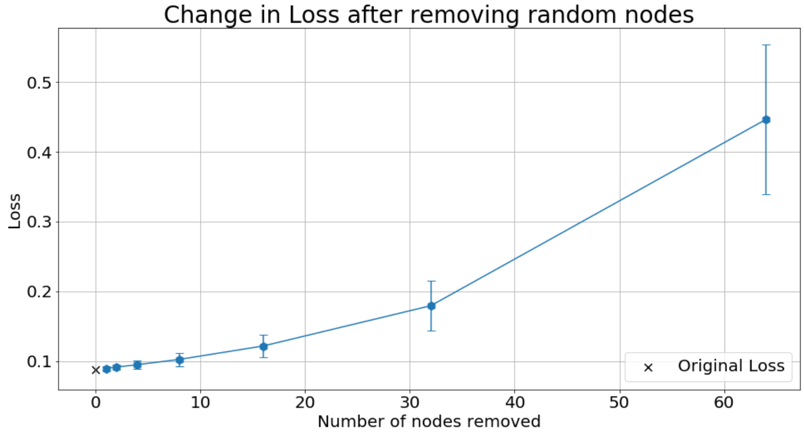

Figure 3.2: Testing

Trial text
Trial text
Trial text
There is trial text here
Trial text
Trial text
Trial text

	1	2	3	4	8
mean	7.4000	8.9000	8.2500	5.8000	2.4000
std	1.6983	2.7891	2.7314	3.3023	3.7613
min	5.0000	4.0000	3.0000	0.0000	0.0000
25%	6.0000	7.5000	6.0000	4.0000	0.0000
50%	7.0000	8.0000	9.0000	4.0000	0.0000
75%	8.2500	10.0000	9.7500	8.0000	8.0000
max	11.0000	16.0000	12.0000	12.0000	8.0000

Table 3.3: Long

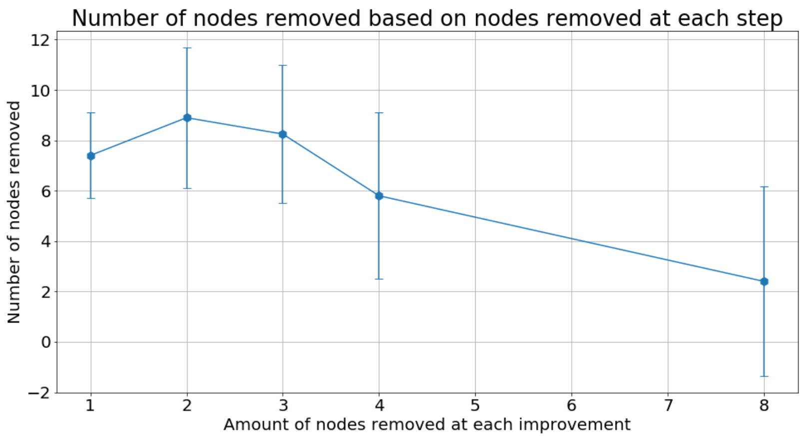

Figure 3.3: Testing

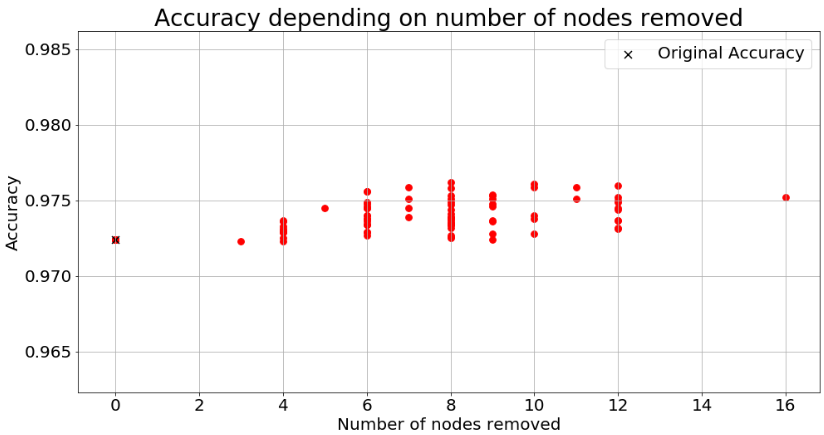

Figure 3.4: Testing

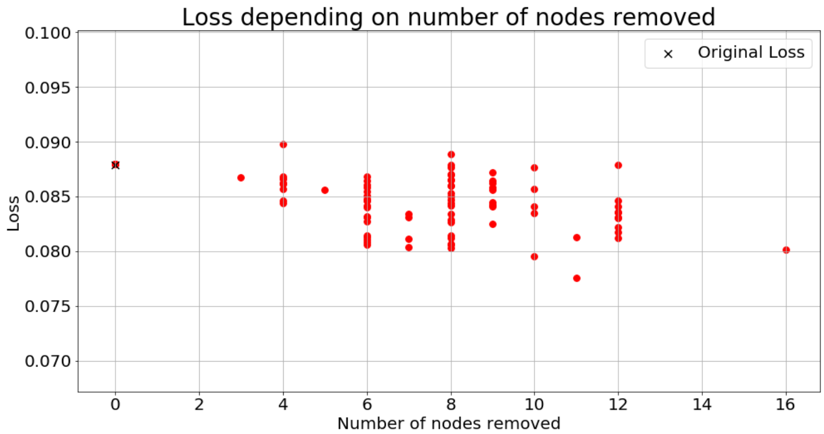

Figure 3.5: Testing

	1	2	4	8	16	32	64
mean	0.8677	0.8667	0.8652	0.8618	0.8543	0.8295	0.7394
std	0.0021	0.0030	0.0042	0.0059	0.0096	0.0200	0.0443
min	0.8578	0.8470	0.8470	0.8405	0.8050	0.7382	0.5493
25%	0.8670	0.8653	0.8632	0.8582	0.8491	0.8190	0.7171
50%	0.8684	0.8674	0.8659	0.8626	0.8558	0.8336	0.7437
75%	0.8686	0.8685	0.8681	0.8663	0.8611	0.8441	0.7698
max	0.8726	0.8735	0.8745	0.8767	0.8747	0.8656	0.8307

Table 3.4: Long

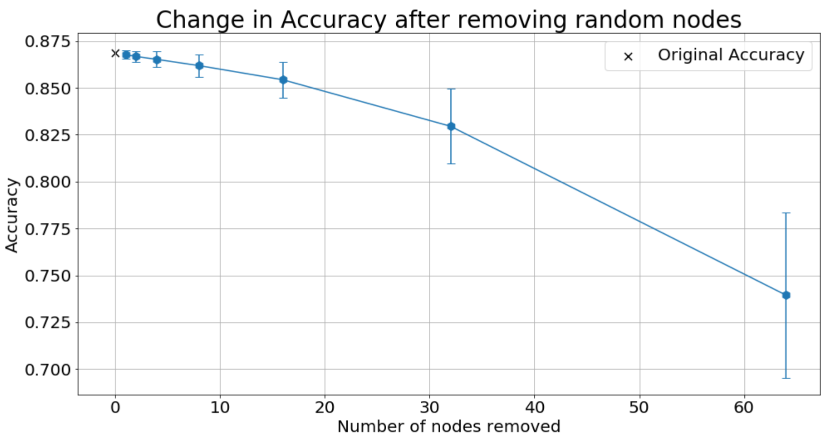

Figure 3.6: Testing

Trial text

	1	2	4	8	16	32	64
mean	0.3578	0.3601	0.3643	0.3730	0.3939	0.4607	0.7062
std	0.0049	0.0070	0.0098	0.0137	0.0228	0.0495	0.1182
min	0.3459	0.3452	0.3434	0.3448	0.3515	0.3725	0.4730
25%	0.3557	0.3559	0.3576	0.3629	0.3772	0.4245	0.6235
50%	0.3564	0.3584	0.3625	0.3711	0.3900	0.4504	0.6867
75%	0.3591	0.3632	0.3693	0.3811	0.4060	0.4857	0.7619
max	0.3791	0.4055	0.4035	0.4297	0.5106	0.7079	1.3649

Table 3.5: Long

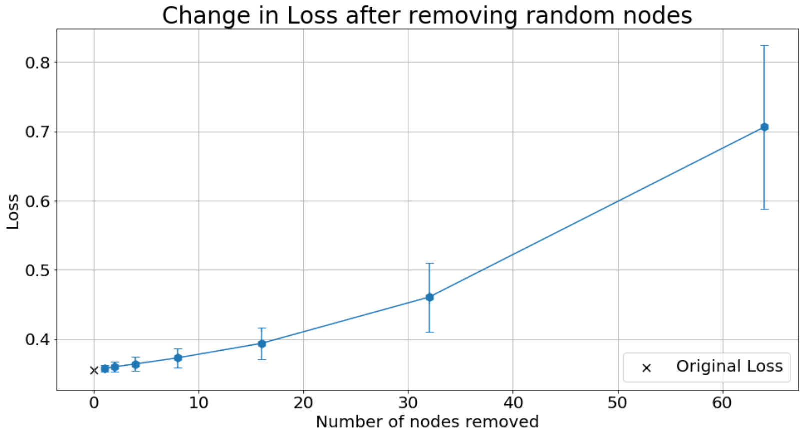

Figure 3.7: Testing

	1	2	3	4	8
mean	5.5000	8.6000	10.6500	8.400	5.6000
std	1.5728	1.8468	3.1502	3.409	3.7613
min	2.0000	6.0000	3.0000	4.000	0.0000
25%	5.0000	8.0000	9.0000	7.000	0.0000
50%	5.5000	8.0000	12.0000	8.000	8.0000
75%	6.2500	10.0000	12.0000	12.000	8.0000
max	8.0000	12.0000	18.0000	16.000	8.0000

Table 3.6: Long

3.2 Estimating Node Importance based on Loss

TODO - Describe the algorithms used

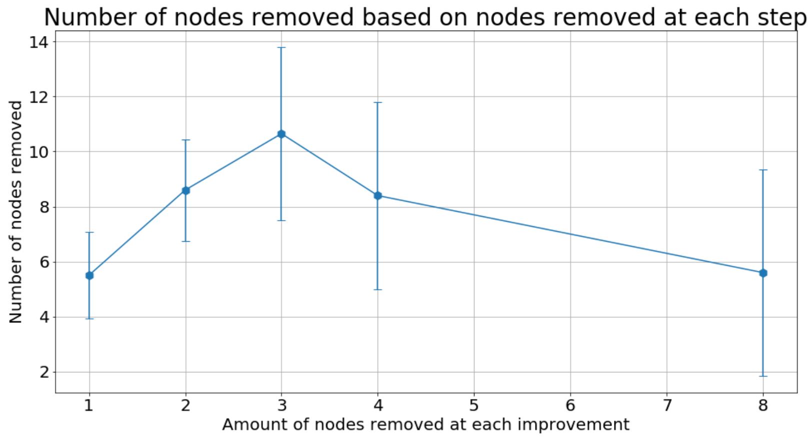

Figure 3.8: Testing

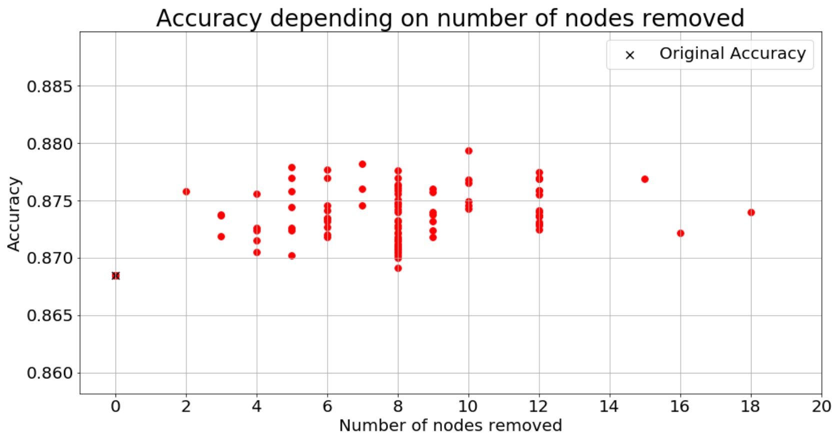

Figure 3.9: Testing

3.3 Pruning network with pre-calculated importance

TODO - Describe the algorithms used

3.4 Pruning Nodes based on the Loss

TODO - Describe the algorithms used

3.5 Greedy approach to pruning instead of Exhaustive approach

TODO - Describe the algorithms used

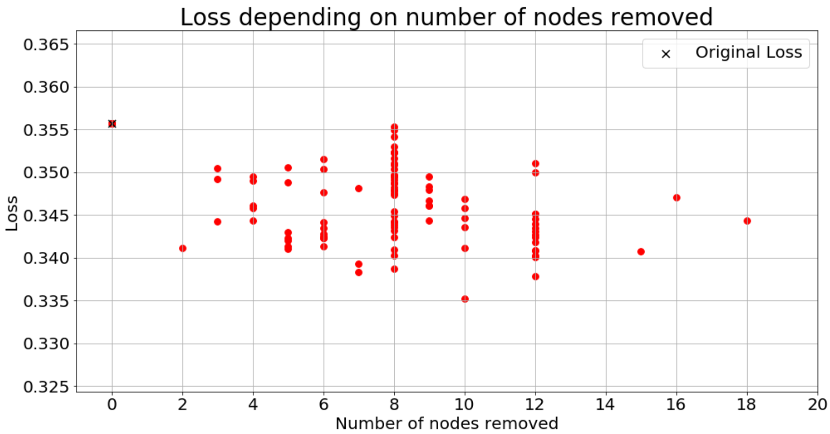

Figure 3.10: Testing

Chapter 4

Multi-Layer Perceptron

4.1 Effects of Changing Training Batch Size on Node Importance

4.2 Effects of Using Dropout

4.3 Iterative weight initialization using Node importance

Chapter 5

Convolutional Neural Network

5.1 Looking at effects of per class accuracy after pruning

5.2 Pruning based on class accuracy

Chapter 6

Case study: Reducing a VGG-16 model trained on X dataset

Chapter 7

Conclusion

7.1 Summary

7.2 Future Works