University of Bergen
Department of Informatics

Machine Teaching using minimzation over complexity of instances

Author: Brigt Arve Toppe Håvardstun
Supervisors: Jan Arne Telle

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

February, 2022

Abstract

In todays society AI and machine learning is becoming more and more relevant. (find source) For instance, some might say the age old problem of the protein folding has finally been solved thanks to Deep mind and their Alphafold 2 (source). Their deep learning approach achieved close to 90% accuray, matching experimental approaches. However, even if we now can get greatly accurat approximation solutions to the question of protein folding, the question of how each protein fold into their 3D structure has not been solved yet. Motivated by this fact; that knowing solutions to instances not nessecery gives insight to the question at large, as well as the expanding use of AI in our everday life [image recognition, recommendation systems, personal medecin and law]. This places Explaineable AI as a topic of of extrem relevancy in the comming years. Our work will regard the topic of model-agnoitic, global, example baed Explaineable Machine Learning. We will focus on the complexity of each instance to be used for teaching, and not on how many examples we are going to use.

Acknowledgements

Est suavitate gubergren referrentur an, ex mea dolor eloquentiam, novum ludus suscipit in nec. Ea mea essent prompta constituam, has ut novum prodesset vulputate. Ad noster electram pri, nec sint accusamus dissentias at. Est ad laoreet fierent invidunt, ut per assueverit conclusionemque. An electram efficiendi mea.

`Brigt Arve Toppe Håvardstun`
Monday 28th February, 2022

Contents

1 Introduction **1**
- 1.1 Background . . . 1
  - 1.1.1 Listings . . . 1
  - 1.1.2 Figures . . . 2
  - 1.1.3 Tables . . . 2
  - 1.1.4 Git . . . 3

2 Model **4**
- 2.1 θai - the AI . . . 4
- 2.2 θLM - modeling human learner . . . 4

3 Dataset **5**
- 3.1 Background . . . 5
- 3.2 Description of an instance . . . 6
- 3.3 Properties of instance . . . 6
- 3.4 Parameters generating instances . . . 7
  - 3.4.1 $FixedSquares$ – placement of letter . . . 7
  - 3.4.2 *Rotation* – orientation of the letter . . . 7
  - 3.4.3 $Scale$ - size of the letter . . . 8

Glossary **9**

List of Acronyms and Abbreviations **10**

Bibliography **11**

A Generated code from Protocol buffers **12**

List of Figures

1.1 Caption for flowchart 2

3.1 Bitmap A 6
3.2 Bitmap AD 6
3.3 Bitmap BCD 6
3.4 Bitmap ABCD 6
3.5 Bitmap ABD 6
3.6 Bitmap CD 6
3.7 Bitmap w/rotation ABC 8
3.8 Bitmap ACD, too big. Overlapping, and out of image. 8
3.9 Bitmap AB, too small. A is broken up, B looks like R. 8

List of Tables

1.1 Caption of table 3

Listings

1.1 Short caption 1
1.2 Hello world in Golang 2
A.1 Source code of something 12

Chapter 1

Introduction

In this master thesis we implement and discuss the reulst of the propsed model for machine teaching. [1]

1.1 Background

Lorem ipsum dolor sit amet, cu graecis propriae sea. Eam feugiat docendi an, ei scripta blandit pri. Nonumes delicata reprimique nam ut. Eu suas alterum concludaturque est, ferri mucius sensibus id sed [2].

We can do glossary for acronymes and abriviations also: Software as a Service (SaaS). As you see the first time it is used, the full version is used, but the second time we use SaaS the short form is used. It is also a link to the lookup.

1.1.1 Listings

You can do listings, like in Listing 1.1

Listing 1.1: Look at this cool listing. Find the rest in Appendix A.1

```
$ java -jar myAwesomeCode.jar
```

You can also do language highlighting for instance with Golang: And in line 6 of Listing 1.2 you can see that we can ref to lines in listings.

Listing 1.2: Hello world in Golang

```
package main

import "fmt"

func main() {
    fmt.Println("hello world")
}
```

1.1.2 Figures

Example of a centred figure

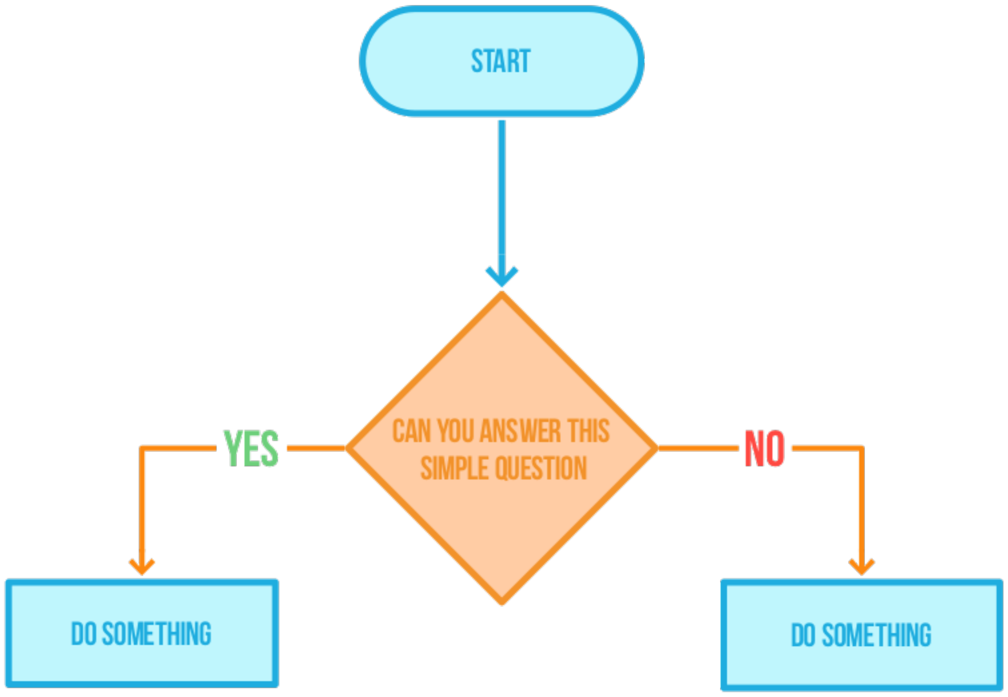

Figure 1.1: Caption for flowchart

Credit: Acme company makes everything `https://acme.com/`

1.1.3 Tables

We can also do tables. Protip: use `https://www.tablesgenerator.com/` for generating tables.

Table 1.1: Caption of table

Title1	Title2	Title3
data1	data2	data3

1.1.4 Git

Git is fun, use it!

Chapter 2

Model

In this thesis there are multiple "models" and we will therefor establish cleare notations and descriptions of of each.

2.1 θai - the AI

With θai we refere to the AI model we are trying to teach to a human. In this thesis we will experimented with two different implementations of θai. One being a convolutional nural network, refered to as CNN, and the other being a fully connected nural network, refrenced to as NN.

2.2 θLM - modeling human learner

Chapter 3

Dataset

In this chapter we discuss how the dataset was choosen, how we create it, and the different parameters acceseable when creating it.

3.1 Background

Our first decision was in desciding the task our AI was going to solve. In selecting the task our AI model was going to solve we had a few attributes in mind. We did not want something to complex, it had to be simple to implement and change, as we were to make a Proof Of Consept. At the same time the problem should not be too simple, we want to check if our method will convey the AI models errors with respect to the ground truth. We also wanted to able to test our system with different AIs. To achive this we wanted our task to achive this we wanted the ground truth to be easily changeable, so we can compare AIs trained on different ground truths.

In the end we chose the task of predicting an underlaying boolean function given bitmaps contaning a subset of the literals A,B,C,D. Present literals in the bitmap is set to True, and all others are set to False. The label for a given instance is then found by calulcateing the underlaying boolean function with literal assigned as descrebied above, giving a label of 1 – being True – or 0 – being False.

This choice fits our demands as we can easily create a new ground truth by swapping the ground truth boolean function. This means our model is not over complex. At the same time the bitmaps representation of literals gives us the possibility of huge traing datasets, which has the possibility of creating more or less complex datasets – different orientations, sizes and letter styles.

3.2 Description of an instance

In more detail, the input to our AI will be a 64x64 bitmap. The cells in the bitmap will either have the value 0 or 255. The cells having the value 255 will together make up one or more letter from a predetermined letter set $L = A, B, C, D$. Three examples of the instances are shown below.

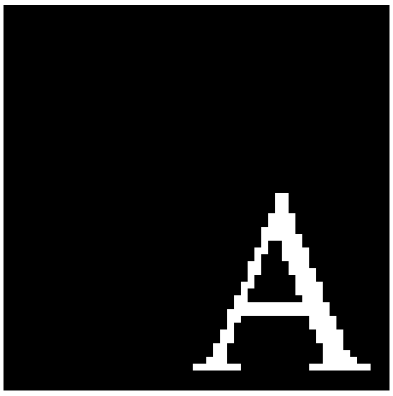
Figure 3.1: Bitmap A

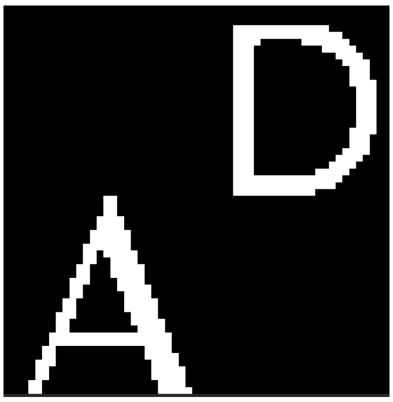
Figure 3.2: Bitmap AD

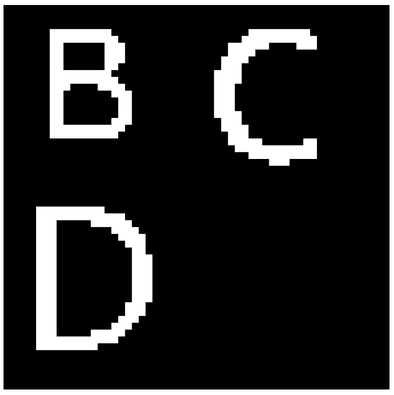
Figure 3.3: Bitmap BCD

We also have different parameters to active. These will be discussed in 3.4. The images above are of the smallest varience possible. Below are examples showing images from our dataset at highest possible varience.

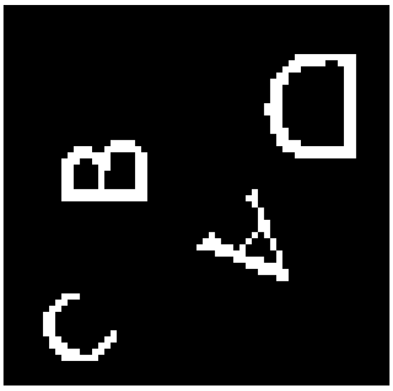
Figure 3.4: Bitmap ABCD

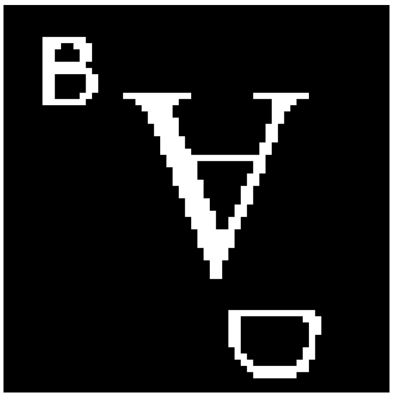
Figure 3.5: Bitmap ABD

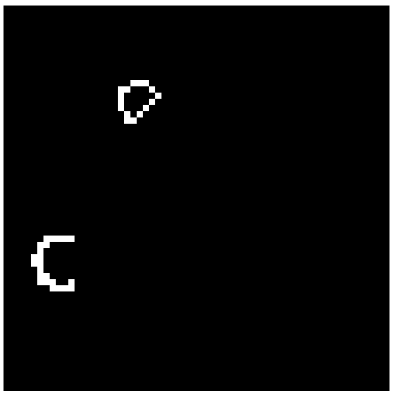
Figure 3.6: Bitmap CD

3.3 Properties of instance

We want to have som clear properties for each data instance. These are mostly sanity checks to make sure our AI will be given decent data. Altough intuitive, its important to make sure our data is sane. Bad data in, means bad perdictions comming out.

Our rules for the data goes as follows:

- For each letter, the entirety of the letter is to be contained within the picture.
- No letter is to overlap with another letter.
- The size of the letter should not be to small. The letter shall contain enough pixles, such that the characteristics of the letter is mainteind. E.g. clearly recognizeable for a human.

3.4 Parameters generating instances

When generating data we have quite a few differtent parameters, all contributing to more or less complexity in our data. The goal of having different parameters for dataset generating is to be able to have a smooth increasing in difficulty for our AI. Using this we can find a suitable dataset for our AI models, where they preforme resonabley well. We can also try to detect different overfittings based on these parameters.

3.4.1 *FixedSquares* – placement of letter

For the parameter *FixedSquares* we divide the image into four squares –top left, top right, bottom left, bottom right. Each of these square might then contain a letter. Crucially no letter is overlapping these squares. All of figures 3.1, 3.2 and 3.3 have *FixedSquares*. Having this parameter turned on will make the dataset less complex, as its counter part is to just place freely. Without *FixedSquares* the algorithm might find all letters at the top of the image or at the bottom, increasing the varience of the dataset. This corresponds to a more complex dataset, less prone to be easily overfitted.

3.4.2 *Rotation* – orientation of the letter

With the parameter *Rotation* we alow the letters to have different orientations. We have slected a subset of orientations $O = \{0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°\}$. Letting each letter take on one of these orientations increase varience and complexity in our dataset. Below is an example of an image from a dataset with *Rotation* allowed.

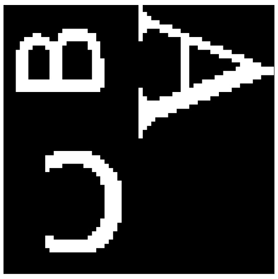

Figure 3.7: Bitmap w/rotation ABC

3.4.3 *Scale* - size of the letter

With *Scale* we allow letters to have different sizes. Importantly no letter can be made to big 3.8, as it has to fit inside the image. Nor can it be made to small 3.9, as it needs to be recognizeable.

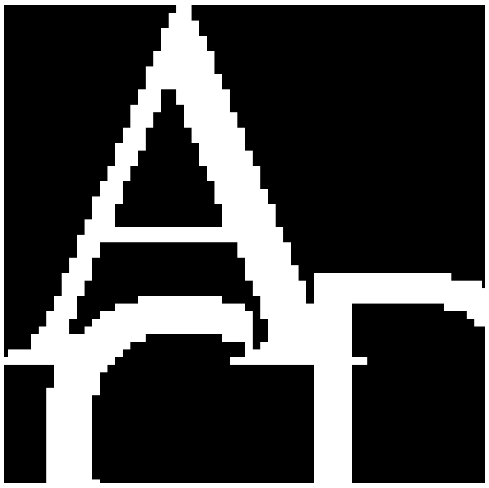

Figure 3.8: Bitmap ACD, too big. Overlapping, and out of image.

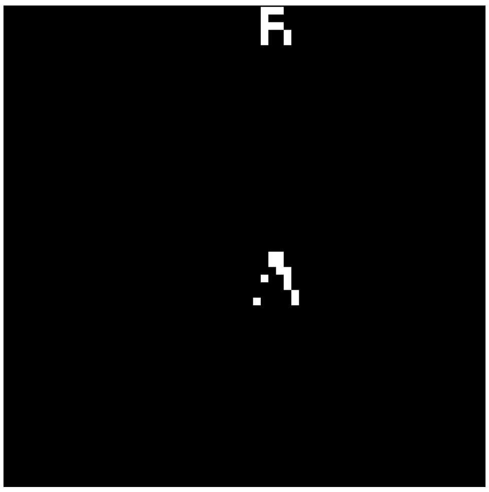

Figure 3.9: Bitmap AB, too small. A is broken up, B looks like R.

Having a broader range of sizes on the images will increase varience and complexity of our dataset.

Glossary

Git Git is a Version Control System (VCS) for tracking changes in computer files and coordinating work on those files among multiple people.

List of Acronyms and Abbreviations

SaaS	Software as a Service.
VCS	Version Control System.

Bibliography

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015. ISSN 1476-4687. doi: 10.1038/nature14539.
URL: `https://doi.org/10.1038/nature14539`.

[2] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, pages 305–320, Berkeley, CA, USA, 2014. USENIX Association. ISBN 978-1-931971-10-2.

Appendix A

Generated code from Protocol buffers

Listing A.1: Source code of something

```
System.out.println("Hello Mars");
```