Oskar Bechtold, B.Sc.

AI Cruciverbalist - Artificial Intelligence (Machine Learning and Constraint Satisfaction Programming) for Grid Based Word Puzzle Generation

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Roman Kern, Dipl.-Ing. Dr.techn.

Institute for Interactive Systems and Data Science
Head: Stefanie Lindstaedt, Univ.-Prof. Dipl.-Inf. Dr.

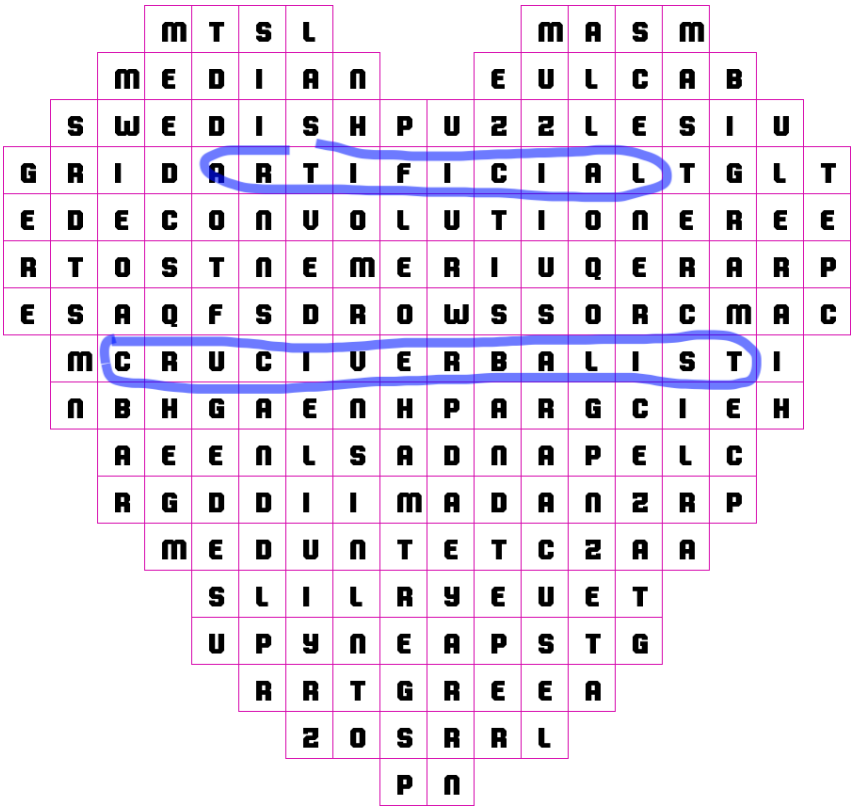
Graz, August 2020

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date	Signature

To all my friends all over the world
Thank you for the love!



Special thanks to Stefan Krüger for introducing me to the puzzle world, Manuel Pistner for being a great mentor and to Roman Kern for supporting me in exploring the exciting and challenging field of data science.

Solution in chapter C

Abstract

This thesis presents a novel way of creating grid-based word puzzles, named the *AI Cruciverbalist*. These word puzzles have a large fan base of recreational players and are widespread in education. The puzzle creation process, an NP-hard problem, is not an effortless task, and even though some algorithms exist, manual puzzle creation achieved the best results so far. Since new technologies arose, especially in the field of data science and machine learning, the time had come to evaluate new possibilities, replace existing algorithms and improve the quality and performance of puzzle generation. In particular *neural networks* and *constraint programming* were evaluated towards feasibility, and the results were compared. The black box of a trained model makes it hard to ensure positive results, and due to the impossibility of modelling some requirements and constraints, neural networks are rated unsuitable for puzzle generation. The significance of correct values in puzzle fields, the approximative nature of neural networks, and the need for an extensive training set additionally make neural networks impractical. On the other hand, precisely modelling requirements for a constraint satisfaction problem has shown to create excellent results, finding an exact solution, if a solution exists. The presented results achieved with the constraint programming approach are rated as successful by domain experts, and the algorithm has been successfully integrated into an existing puzzle generator software for use in production.

Contents

List of Figures

1. Introduction

“A good puzzle makes the solver feel smart.“

- David Kwong

1.1. Introduction

"Crosswords are the shit" as Schreiber-Stainthorp [151] says in his analysis of the history of crosswords. The field of puzzles, sweepstakes, and horoscopes, as commonly used in newspapers, magazines, and mobile apps, has a large fan base of regular and occasional players. Not only recreationally for pure entertainment, but also in education to study languages and concepts. The business opportunities around these fields generate significant revenues and create many jobs. However, it is a very challenging business. Creation of crossword puzzles and word games, for example, is not an effortless task: it is often done manually or semi-automatically. The semi-automatically created puzzles then need manual improvement by an expert editor. This process is time-consuming and expensive. As stated in *How to Make a Crossword Puzzle: Detailed Instructions for Beginners* [86] "The first thing you should know about making a newspaper-quality crossword puzzle is that it’s tough! Creating a good puzzle can be more challenging, and more fulfilling, than solving one." The puzzle generation process is an NP-hard problem, and even though some algorithms that automatically generate word-based puzzles do exist, the quality is often considered worse than manually created puzzles. Also, not all puzzle types can be generated automatically yet, at least

not efficiently. With new technologies arising, especially in the field of data science and machine learning, the time has come to replace existing algorithms and improve the quality and performance of puzzle generation, with new algorithms. In this master thesis, the different stages during the puzzle creation process are examined with a view to improving them with artificial intelligence, deep learning, constraint satisfaction programming, and other data science methods.

1.1.1. What is a Puzzle?

To use Chris Crawford's words in *Game Design Workshop*: "Puzzles are rule-based systems, like games, but the goal is to find a solution, not to beat an opponent. Unlike games, puzzles have little replay value." [74]. This thesis focuses on grid-based word puzzles, especially word search puzzles, explained in more detail in Section 1.2 "Puzzle Types".

1.1.2. Puzzle Creation Stages

There are multiple stages in the process of creating puzzles. As explained in the *New York Times*, there are two steps. The first is creating a theme and selecting words for the theme, as described by Tausig and Vigeland [170]. Then in step two, Steinberg and Last [166] use the themed entries to set them in a blank crossword grid and place the black squares using the software tool *CrossFire* [40]. All stages can be researched, implemented and evaluated individually and are thus split into the following, more detailed, steps for this research.

1. Texts and Vocabularies
    a) Maintaining (themed) word lists
    b) Maintaining (themed) clue and answer lists
    c) Maintaining blacklists of words and clues that should not be used together

   d) Writing content like horoscopes
2. Generating Puzzles
   a) Generating grids with empty word positions
   b) Filling empty grid positions with words
   c) Assigning clues to question fields and corresponding words
3. Quality assurance
   a) Quality of chosen questions/clues/words used in a puzzle
   b) Degree of fun and difficulty concerning
      - Vocabularies
      - Puzzle grids
   c) Is a puzzle solvable at all
   d) Is a puzzle solution deterministic
   e) Word Arrangement
      - Blacklists
      - Orientations
      - Field population
      - Word intersections

Each stage mentioned has its challenges that can be handled with different algorithms, heuristics, and approaches. For this thesis, an extensive data set of horoscopes, puzzles, and vocabularies in the German language is available to be examined. With different data science and deep learning methods, the hidden knowledge will be extracted, and techniques to reuse this knowledge are researched while the overall goal is to generate and compile puzzles.

Initially targeted outcome:

1. A neural network to generate and suggest horoscope texts
2. A puzzle algorithm
   a) A meta-algorithm that checks quality between sub-algorithms and backtracks to redo steps if intermediate quality goals are not met.
      i. A neural net to create puzzles or puzzle grids.
      ii. A constraint satisfaction programming based approach to fill puzzle grids with a given vocabulary.

   iii. A *constraint satisfaction programming* (CSP) based approach to fill puzzle grids with questions and clues.
   iv. An algorithm to check the quality of the generated puzzles.
3. An algorithm to create (themed) word lists, clue, and answer lists.
4. An algorithm to create blacklists.
5. An easy-to-use microservice architecture to integrate the above algorithms into the workflow of puzzle agencies.

The focus is set on puzzle generation, especially on creating and filling grids. While less emphasis is placed on maintaining vocabularies, the vocabulary process might need improvement to increase the overall puzzle quality. Also, while not the focus of attention, quality assurance cannot be skipped, as the whole generation process should spawn high-quality puzzles. Thus the quality of generated puzzles needs to be checked. Wrapping the entire algorithm into an API that is available via a REST will be done to integrate the new algorithms into existing front-end software. In this thesis, two puzzle types, word search (with solution word) and Swedish crossword puzzles are used as a starting point to develop the algorithms and model the data. Once the algorithms can generate these two puzzle types, adaptations to the algorithms for other puzzle types will be made in future work.

1.1.3. Considered Technologies

The technologies listed in this chapter initially gave a frame to limit the field of research. During an early brainstorming session, these technologies seemed to be possible candidates for solving the challenges in the problem space, but not all of them have been used or evaluated.

1. Machine Learning (ML)
   a) Deep Learning (DL)
   b) Reinforcement Learning (RL)

2. Neural Networks
    a) Recurrent neural network (RNN)
    b) Long short-term memory (LSTM)
    c) Gated recurrent unit (GRU)
    d) Variational autoencoder (VAE)
    e) Generative adversarial network (GAN)
    f) ProGAN and StyleGAN
3. Recommender Systems
    a) Content-based filtering to suggest words while filling puzzle grids
4. Constraint Programming (CP)
    a) Constraint Satisfaction Problem (CSP)
    b) Constraint Optimisation Problem (COP)
5. Word Embeddings and Text Classification
    - word2vec and GloVe (context-independent, just a simple vector per word, therefore susceptible to word ambiguities)
    - ELMo and BERT (multiple embeddings per word by respecting words context)
    - Support Vector Machine (SVM)
6. Tools and Libraries
    - Solr / Lucene
    - Tensorflow / Keras
    - Google Optimization Tools (OR-Tools)

1.1.4. Available Data

An extensive puzzle database is available for examination, for training models, and for evaluating the structure of existing puzzles, courtesy of Krupion GmbH.

- One million PDFs containing multiple puzzles and horoscopes full text searchable via Solr.
- At least 2000 regular, high-quality puzzles in XML format.

- Vocabularies, including different levels of difficulty, with 70000 words.
- 32000 Blacklists, with about 100 excluded words each. Parts of these are automatically created, so they are not always of the highest quality.

1.2. Puzzle Types

"Variety's the very spice of life, that gives it all its flavor."

- William Cowper

A vast amount of dedicated word puzzles to study or test language levels, logical thinking or question-answer tests do exist. This study focuses on grid-based puzzles. While only a few examples are listed here, more than 50 grid-based word puzzle types exist. With the methods and technologies evaluated, it is expected to be possible to create the following puzzle types in a very efficient way and to the highest quality, as the data structure of the puzzle types is very similar and adapting the algorithms is a small transfer task.

- Swedish puzzle as seen in Figures 1.1a and 1.2b
- Swedish puzzle without vowels (some vowels are prefilled)
- Swedish puzzles with words wrapping around its edges
- Swedish puzzles with unusual shapes (for example hearts) as seen in Figure 5.7
- Classical crossword puzzles as seen in Figure 1.1c
- Grid puzzles as seen in Figure 1.1d
- French, Swiss, and Norwegian puzzles
- Tricky combination: words placed horizontally to form a diagonal solution
- Word search as seen in Figure 1.1b
- Word search with solution word
- Syllable based crossword as seen in Figure 1.2a

- Bridge puzzle
- Puzzle quartet
- Honeycomb puzzle as seen in Figure 1.1e
- Snail puzzle as seen in Figure 1.1f
- With small adaptions any puzzle type presented in the Rätsel Krüger GmbH catalogue[1] is possible to be generated with the proposed sets of algorithms.

The following two puzzle types, letter desert / word search (with solution word) and Swedish crossword puzzles are the focus while developing the algorithms. The other grid-based puzzle types are only considered as inspiration for developing a universal data model but are left out for future work.

1.2.1. Letter Desert

In letter desert puzzles - often also called "word search puzzles" - terms are hidden in an array of letters. The words can run in any direction, vertical, horizontal, diagonal, and backward. Words can intersect in shared letters and can relate to a specific theme to create a thematic puzzle. The goal is to find all the hidden words. Optionally, a list of the words hidden in the puzzle grid is provided. This gives the player a clue as to which words to search for. An example can be seen in Figure 1.1b.

1.2.2. Word Search

A word search puzzle is a specialization of the classic letter desert puzzle. The same rules apply as for the letter desert, but the puzzles are enriched with a solution word. In this puzzle type, when all words are found, the leftover fields reveal a solution word. The goal is to find all hidden words to reveal the solution word letters. The solution

[1] https://www.krupion.de/krupion_frontpage/files/kataloge/Krupion_Raetselkatalog_Klassisch.pdf (Accessed on 2020-07-13)

word letters can be in order or be randomised to create the additional task of solving the solution word. This thesis focuses on word search puzzles as defined here to limit the research area. An example can be seen in Figure 1.1b.

1.2.3. Crossword

A crossword puzzle is a word search game usually in the form of a rectangular or square grid of white and black squares. The goal of crossword puzzles is to fill the empty white squares with letters, forming words, by solving clues. Answering these clues reveals the words to be filled into the grid. Black squares separate the words or phrases from each other. Clues listed beside the puzzle grid are numbered to indicate where the answer needs to be placed in the grid. Often the clues are split into horizontal and vertical word lists to give the player an indication of the direction of answers. Crossword puzzles are usually rotationally symmetrical. Vocabularies can be made up of generic words from arts, history, science, and others, and can be specialized and themed with words from a specific topic. An example of a crossword puzzle can be seen in Figure 1.1c.

1.2.4. Swedish Puzzle

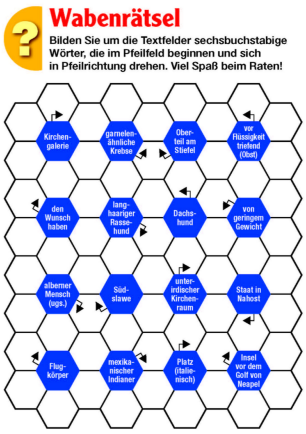
The Swedish puzzles grid does not have clue numbers, as the clues are contained in the black cells within the grid. In addition to the clue fields, arrows indicate in which position and in which direction answers have to be filled in: vertical or horizontal. This puzzle style is the most favoured puzzle type in magazines and daily newspapers in German-speaking regions. Usually, the Swedish puzzles do not have a symmetrical grid. An example of a Swedish puzzle can be seen in Figure 1.1a.

(a) Swedish Puzzle

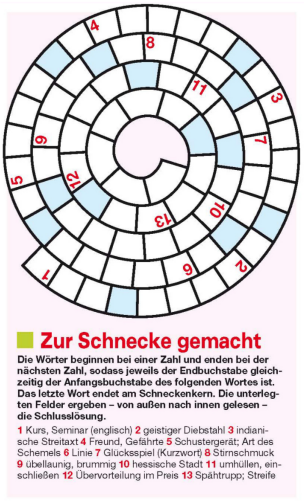
(b) Word Search Puzzle

(c) Crossword Puzzle

(d) Grid Puzzle



(e) Honeycomb Puzzle

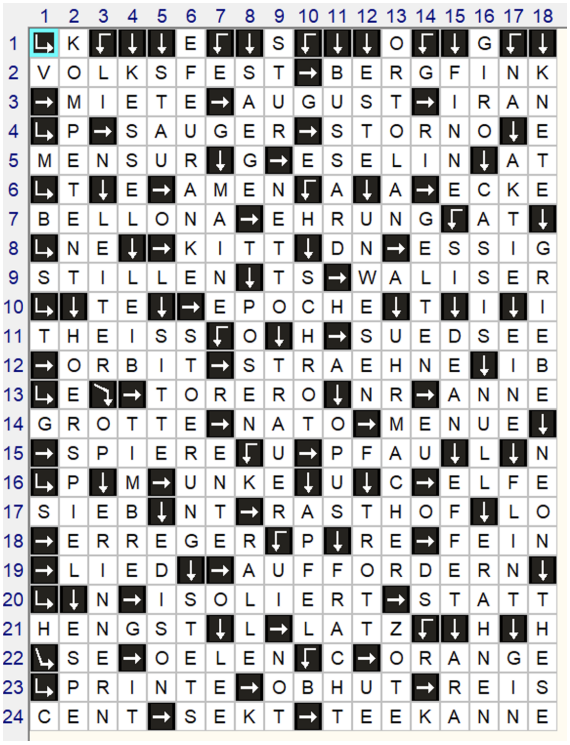


(f) Snail Puzzle

Figure 1.1.: Puzzle Type Examples (Source: Krupion GmbH, illustration with friendly permission, 2020)

	1	2	3	4
1	UN	KE	■	WA
2	RU	■	LU	RE
3	HE	LE	NA	■
4	■	GA	■	LA
5	AU	TO	MO	BIL
6	LA	■	BI	■
7	■	SPIE	LE	REI
8	I	GEL	■	BE
9	■	EI	RE	■
10	MI	■	TI	DE
11	DI	A	NA	■
12	■	ME	■	RO
13	O	RI	NO	KO
14	■	KA	■	KO
15	KEI	NER	LEI	■
16	FEN	■	SE	MI

(a) Raw Syllable Puzzle



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	■	K	■	■	■	E	■	■	S	■	■	■	O	■	■	G	■	■
2	V	O	L	K	S	F	E	S	T	■	B	E	R	G	F	I	N	K
3	■	M	I	E	T	E	■	A	U	G	U	S	T	■	I	R	A	N
4	■	P	■	S	A	U	G	E	R	■	S	T	O	R	N	O	■	E
5	M	E	N	S	U	R	■	G	■	E	S	E	L	I	N	■	A	T
6	■	T	■	E	■	A	M	E	N	■	A	■	A	■	E	C	K	E
7	B	E	L	L	O	N	A	■	E	H	R	U	N	G	■	A	T	■
8	■	N	E	■	■	K	I	T	T	■	D	N	■	E	S	S	I	G
9	S	T	I	L	L	E	N	■	T	S	■	W	A	L	I	S	E	R
10	■	■	T	E	■	■	E	P	O	C	H	E	■	T	■	I	■	I
11	T	H	E	I	S	S	■	O	■	H	■	S	U	E	D	S	E	E
12	■	O	R	B	I	T	■	S	T	R	A	E	H	N	E	■	I	B
13	■	E	■	■	T	O	R	E	R	O	■	N	R	■	A	N	N	E
14	G	R	O	T	T	E	■	N	A	T	O	■	M	E	N	U	E	■
15	■	S	P	I	E	R	E	■	U	■	P	F	A	U	■	L	■	N
16	■	P	■	M	■	U	N	K	E	■	U	■	C	■	E	L	F	E
17	S	I	E	B	■	N	T	■	R	A	S	T	H	O	F	■	L	O
18	■	E	R	R	E	G	E	R	■	P	■	R	E	■	F	E	I	N
19	■	L	I	E	D	■	■	A	U	F	F	O	R	D	E	R	N	■
20	■	■	N	■	I	S	O	L	I	E	R	T	■	S	T	A	T	T
21	H	E	N	G	S	T	■	L	■	L	A	T	Z	■	■	H	■	H
22	■	S	E	■	O	E	L	E	N	■	C	■	O	R	A	N	G	E
23	■	P	R	I	N	T	E	■	O	B	H	U	T	■	R	E	I	S
24	C	E	N	T	■	S	E	K	T	■	T	E	E	K	A	N	N	E

(b) Raw Swedish Puzzle

Figure 1.2.: Puzzle Examples of Raw Puzzles (Source: Krupion GmbH, illustration with friendly permission, 2020)

1.3. Vocabulary Types

Vocabularies consist of words and questions which are passed to the algorithm to create puzzles. Different types of vocabularies serve various tasks in the generation process. These types are described here.

- **Standard word vocabulary**, as seen in Figure 1.3a, with general terms.
- **Themed vocabulary**, as seen in Figure 1.3b, with words that fit together thematically.
- **Word-question vocabulary** for Swedish puzzles, as seen in Figure 1.3d. It maps questions and clues to words (can be multiple clues per word). For example, the questions "Software on your phone" and "Short for application" could be assigned to the word "APP".
- **Synonyms**, as seen in Figure 1.3c.
- **Blacklists** that exclude words from being used together.

Different Types of Vocabularies.

```
A
AA
AAL
AALEN
AARON
AB
ABO
ABRAUM
AD
ADE
ADEL
AESTHET
AFFE
AG
AHA
AI
AKT
ALE
ALI
ALLESKLEBER
ALOE
ALPEN
ALT
ALTE
AM
AMADEUS
AMOEBE
AN
```

(a) Standard Vocabulary

```
ABBA
ALPHAVILLE
BACKSTREETBOYS
BEASTIEBOYS
BEEGEES
BLONDIE
CHICAGO
COLDPLAY
DURANDURAN
ERASURE
KARAT
KRAFTWERK
MADSEN
NAZARETH
NSYNC
OASIS
PUHDYS
QUEEN
ROXETTE
SCORPIONS
THEBEACHBOYS
THEDOORS
TOTO
URIAHHEEP
ZZTOP
```

(b) Band Themed Vocabulary

```
AASEN=verschwenden|ver-#schwen-#den
AASEN=vergeuden|vergeu-#den
AASEN=verprassen|verpras-#sen
ALB=Alm
ALB=Alp
ALB=Alpe
ALB=Matte
ALB=Alpweide|Alpwei-#de
ALB=Bergweide|Bergwei-#de
ALB=Bergwiese|Bergwie-#se
ANEMONE=Windröschen|Windrös-#chen
ANGINA=Mandelentzündung|Mandel-#entzün-#dung
ANKLANG=Vorgeschmack|Vorge-#schmack
ANKLANG=Vorzeichen|Vorzei-#chen
ANKLANG=Sympathie|Sympa-#thie
ANKLANG=Zustimmung|Zustim-#mung
ANKLANG=Affinität|Affi-#nität
ANKLANG=Reminiszenz|Remi-#niszenz
ANREDEN=adressieren|adres-#sieren
ANREDEN=ansprechen|anspre-#chen
ANSTIEG=Zunahme
ANSTIEG=Zuwachs
ANSTIEG=Mehrung
```

(c) Synonyms vocabulary

Other Types of Vocabularies.

```
AASEN=verschwenden|ver-#schwen-#den
AASEN=vergeuden|vergeu-#den
AASEN=verprassen|verpras-#sen
ALB=Alm
ALB=Alp
ALB=Alpe
ALB=Matte
ALB=Alpweide|Alpwei-#de
ALB=Bergweide|Bergwei-#de
ALB=Bergwiese|Bergwie-#se
ANEMONE=Windröschen|Windrös-#chen
ANGINA=Mandelentzündung|Mandel-#entzün-#dung
ANKLANG=Vorgeschmack|Vorge-#schmack
ANKLANG=Vorzeichen|Vorzei-#chen
ANKLANG=Sympathie|Sympa-#thie
ANKLANG=Zustimmung|Zustim-#mung
ANKLANG=Affinität|Affi-#nität
ANKLANG=Reminiszenz|Remi-#niszenz
ANREDEN=adressieren|adres-#sieren
ANREDEN=ansprechen|anspre-#chen
ANSTIEG=Zunahme
ANSTIEG=Zuwachs
ANSTIEG=Mehrung
```

(d) Word-question vocabulary including line breaks

Figure 1.3.: Examples of basic and themed word vocabularies, synonym vocabularies and word-question vocabulary used for the puzzle generation process. (Source: Krupion GmbH, 2020)

2. Background

> *"I am just a child who has never grown up. I still keep asking these 'how' and 'why' questions. Occasionally, I find an answer."*
>
> - Stephen Hawking

Why do we need crossword puzzles? What makes them unique? Moreover, what keeps thousands of people playing these puzzles? Already in the roaring 1920 "young people also spent their free time working crossword puzzles" as stated by Edge [61, p. 27]. The first crossword puzzle, created by Arthur Wynn appeared in *The New York World* on December 21, 1913 [50]. A lot has changed since then. Many new puzzle types were invented, critical voices questioned the community [101], the use of puzzles extended beyond pure leisure and puzzle creation supported by computers was researched.

This chapter takes a look at related work, applications of word puzzles, and existing research concerning the generation of puzzles. First the use cases of puzzles are examined and a wide distribution in teaching and a strong recommendation for language learning will be seen. Solving puzzles as tasks during experiments has also been used to do research on cognition and human behaviour. While some proprietary software and some patents exist only few requirements introduced in Chapter 3.2 could be fulfilled. A deeper look is taken into scientific research focusing on the puzzle creation process and the process of solving puzzles. Most research found used genetic algorithms or constraint programming focusing on American style puzzles and Sudoku puzzles. Many research papers show a separation of tasks

like creating vocabularies, generating a grid and filling it. The results of the separation can be summarized to improve the performance of the introduced algorithms. While data mining and natural language processing was used sometimes to create puzzle vocabularies or to solve crosswords, no research using neural networks for the puzzle generation process was published so far.

2.1. Related Work

> *"We cannot solve our problems with the same thinking we used when we created them."*
>
> - Albert Einstein

Higashida [84] argues that computers changed the puzzle world by improving the creation process of, for example, arithmetical puzzles, cryptic puzzles, mechanical puzzles and paper puzzles like Sudoku. Puzzles nowadays, created with the support of computers, are "more beautiful".

Only a few tutorials on the puzzle creation process aimed at non-scientists [87] exist and usually describe the creation of American style puzzles. Mike Vuolo [117] explains the rotational symmetry, often called *crossword symmetry*, used in *New York Times* puzzles. A bit more detail is explained by Kwong [99] a New York Times puzzle constructor. The complexity of the creation process makes the description vulnerable to errors that need clarification, for example, by Matt Gaffney [110] who explains "How Crossword Puzzles Are *Really* Made". Focusing on the versatile creation of cryptic crossword clues, a very detailed introduction is given by Michael Callaghan [116].

2.1.1. Teaching

Angel C. de Dios [8] has a sceptical view on using word search puzzles in education as they do not teach being friendly people but only some basic concepts. He claims that no scientific work proving the thesis that using word puzzles do relate to the students' knowledge and that the usage of these puzzles is an "utter waste of time and resources".

On the other hand, Little [105] is quite optimistic and says "Crossword puzzles offer many possibilities for language learning. They can be used to practice and review vocabulary, orthography, morphology, grammatical structures, abbreviations and culturally-specific facts, as well as to develop communicative skills. Given their versatility and adaptability for many different purposes and levels, crossword puzzles have a legitimate place in second language teaching and learning." Also, as stated on *Benefits of Active Recall* [16] quizzes and word puzzles are among the best ways to practice as they make use of active recall. And although Childers [32] can not draw scientific conclusions, he found that solving grid-based crossword puzzles helped students in pulling together information and understanding concepts. This is argued to be happening due to students thinking more about what they were learning.

Davis, Shepherd, and Zwiefelhofer [52] show that it depends on the classes if using crossword puzzles as a 'fun' review technique positively affects successful learning. So a closer look into the different fields of education is taken in the following. Lin and Dunphy [104] proposed using crossword puzzles for business students learning economics. To test their thesis, they developed an experiment with students using crossword puzzles. "The findings revealed that working crossword puzzles can help students build and maintain an introductory microeconomics vocabulary. Test results show that crossword puzzles do in fact aid student learning of microeconomics terms and do improve their grades on quizzes." American style puzzles that follow the rules of connectivity, symmetry and three+ were used by Ferland [71]. He argues that for a 15 x 15 grid, following these three rules, a maximum number of 96 clues is possible. While

students rated the fun of solving crossword puzzles higher than their usefulness, they also explained that the puzzles were helpful in areas of terminology, spelling and definitions. Students who rated the puzzles as not useful found them too hard, too meaningless, or they were not interested in doing them. A comparison of a card game and a crossword puzzle as a teaching aid was made by Franklin, Peat, and Lewis [72]. They found a more robust positive response to the use of crossword puzzles while card games still showed a positive result. Berry and Miller [21, 20] suggest that athletic training educators should use crossword puzzles to improve students' studies as they not only are fun, but they also increase confidence in test-taking and comprehension of the study material. Wise [179, 178] used web-based word puzzles and web-based crossword puzzles as "self-assessment material for student revision" and has shown that "Students' attitudes after using the program were generally positive and students agreed with the proposition that lecturers should be encouraged to use crossword puzzles to support learning." Shah, Lynch, and Macias-Moriarity [155] conclude that "Students perceived that crossword puzzles enhanced their learning of anti-ulcer agents" (drugs and medication). "The use of crossword puzzles provides a simple and creative way to incorporate active learning into pharmacy classroom instruction." Serna [154] let her students solve crossword puzzles, and additionally made them create puzzles as an exercise. While not only "Most students considered this activity as enjoyable and useful for learning." they also found in their study of 84 students that "students' perceived value of crosswords for their learning is positively correlated with both their perceived learning in the course and their attitude toward the course." Also, Weisskirch [176] has shown that students rate the use of crossword puzzles generally favourably as an exercise method. While looking at crossword puzzles solved in class and used at home he found that students "described learning more and finding the exercise more helpful for learning concepts and preparing for the exam when completing the crossword puzzle collaboratively in class rather than individually outside of class. Students also rated the exercise higher as a review technique and as more enjoyable when completed in class." Crossman and Crossman [41] explain that in some classes, for example, the History of Psychology,

relationships, events and sequences need to be learned to pair up some concepts. While they recommend not to draw conclusions or generalise the result, their study participants enjoyed the grid puzzles used and their motivation to study increased. Also, they were able to retain the learned material for longer. They conclude that crossword puzzles seem to be a useful tool for pairing concepts with other information, like names and ideas, for example.

Studying Languages

Using word-based puzzles seems to help to study concepts and to learn words. One of the most significant fields of research is the learning of foreign languages. The most crucial task when studying new languages is memorising vocabularies. As seen before, active recall, which is one of the main activities of studying vocabularies, has positive effects when using word puzzles. Awareness for the pedagogical need is necessary to modify the puzzles accordingly and choosing the right method and puzzles is important says Kunz [98]. This can also be seen in the findings by Sukstrienwong and Vongsumedh [169] who have shown that using word search games in English language learning of students at Bangkok University, Thailand, had improved their studies. Students needed additional sources, however. For example, they used an English Dictionary to learn the pronunciation of the words additionally. Already in 1925, Broome [26] used crossword puzzles for elementary Spanish classes where given clues were English terms and the translated Spanish words needed to be filled into the grid. Also, Buttner [27] suggests using vocabulary puzzles to recall vocabulary by matching words from one to the other language. Improved students‘ achievements in vocabulary in fifth grade in an Indonesian school were found by Ria Damayanti H. [142] who also evaluated using word search puzzle games to learn English vocabulary. They concluded that these puzzles make a suitable technique. Chesy, Susilawati, and Bunau [31] have shown that word search puzzles had improved students’ mastery of vocabulary and rate that word search puzzles are a highly effective method in teaching vocabulary to tenth-grade students.

Educational resources for people learning the Hawaiian language was published by Frayer-Luna [73]. They feature 40 crossword puzzles and eight word search puzzles with some word lists and a Hawaiian language puzzle dictionary with 12000 words. Garba [75] researched using interactive crossword puzzles to teach the English language to speakers of Arabic distinguishing in morphological and syntactic meaning. His results show a positive effect. Another study in Thailand by Orawiwatnakul [128] found that using crosswords in teaching the English language to Thai students improved the vocabulary of the students. Also, the motivation to learn English was increased by providing this fun teaching method. Schafer and Behymer [150] claim that crosswords can help vocational students learning computer skills while increasing technical vocabulary, reinforce spelling and using reference material. Hung and Young [89] built a handheld device for Taiwanese students to learn English vocabulary, hoping to increase students' learning in an interactive environment.

Merkel [115] evaluated using crosswords for English language learners. He found that participants were frustrated by cultural components and context-less nature of regular daily crossword puzzles. They felt more positive and made better progress when working collaboratively. This supports the thesis that puzzles and methods need to be adapted to the teaching goals. McElroy and Samaniego [112] offer crossword puzzles for language learning as an innovative method. They give clear directions for developing puzzles specifically designed to teach grammar, vocabulary, and culture. Different from the others Jones [93] does not use the active recall component of solving puzzles to train vocabulary like many others. He says that "crosswords have considerable potential for the provision of alternative assessment/self-assessment activities".

Many of the studies listed here rate the improved education based on perceived improvements, asking students with questionnaires how they felt. Only a few take the actual test scores into account. Collins [37] describes the need for stimulating students' learning and the importance of teachers searching for new tools. He uses crossword puzzles to activate the quest for knowledge. He had shown that a control group (B) that got crossword puzzles two weeks before

a test outperformed the control group (A) that did not study with the crossword puzzles. While all students in group B have gained better results, only 37% claimed that it stimulated interest is the topic, but 87% agreed that these puzzles are a helpful tool that should be used more often. Mshayisa [122] found that students in *food science and technology* found crossword puzzles required them to think critically. Also, a positive relationship between student collaborative learning and crossword puzzle implementation scores was seen. Using tests before and after using crossword puzzles in eight sessions, Hossein and Marzieh [85] researched the vocabulary learning of Iranian women. They carefully selected 60 homogenous students, split them into two groups and "It was concluded that the subjects who received word-search-puzzle game on vocabulary, outperformed the control group."

Not only solving puzzles in class was found helpful, but creating crossword puzzles as a teaching method found its way to classrooms as well. Nicol [126] found that assigning students with the task of creating a crossword puzzle generated more discussion and pleasure then when solving a given puzzle. While developing the clues to describe words, students seemed to gain more knowledge about a topic and a deeper understanding of concepts. This also promoted discussions, as students are well-versed in a topic after completing the task. Similarly, Jaramillo, Losada, and Fekula [91] examined the creation and solving of crossword puzzles by undergraduate students and conclude that the students found themselves better equipped to handle concepts.

To summarise the mainly positive results, it can be concluded that students using crossword and word search puzzles as teaching aids perform better in their studies. The selection of puzzle type and the pedagogical method need to be aligned to gain the desired benefits successfully. A variety of tasks like finding answers to clues, translating words and creating puzzles exist that can raise students interactivity and fun, as well as encourage active recall to memorise vocabularies and concepts.

2.1.2. Psychological Experiments

Puzzles also found their way into other fields of research, for example as psychological tests researching human behaviour. Opposite conclusions, for example, how to create puzzles for a specific target group, can be drawn from the results. Ghinea and Ademoye [77] have shown that olfactory media (odours, smells, and scents) has a positive impact on solving word search puzzles. During the tests, scents relating to the searched word were given as olfactory clues. Smells grouped the associated words; for example, the strawberry scent was used to give a clue for the words "Fruity", "Strawberry", and "Sweet". The results show positive impacts on solving puzzles when presented with olfactory clues. Alghani, Sutarsyah, and Nurweni [6] and Alghani [5] have used crosswords in their research on trying to find differences in extroverts and introverts vocabulary size using different learning techniques. Similarly, the differences in puzzles types, searching given words vs answering clues, can be seen in the work by Postman, Jenkins, and Postman [136]. Their research focused on active recall and recognition by comparing them. This also indicates which puzzle type to select for the teaching goal. Jessie Chin et al. [92] used word search puzzles of varying difficulty to examine age differences in information foraging. They evaluated how quickly and how often people would switch puzzles. Switching, in general, was more likely under challenging conditions. Especially younger participants switched more often than older ones. Younger people also showed faster uptake, which was less predictive, however. "Older people persevered longer, especially in more difficult condition." Day [53] made an experiment researching the different perceiving of language. They found that "stimulus-bound" (SB) subjects consistently found more words in word search puzzles than "language-bound" (LB) subjects. This might be due to the thesis that SBs have better spatial abilities which help in scanning the grid in eight directions. Another possible explanation could be that LBs are preoccupied with linguistic operations. Clayton, Leshner, and Almond [35] used word search puzzles as a cognitive task to examine cognition, emotion and physiology. Zentall, Hall, and Lee [188] used word search puzzles to

investigate if mirrors focus the attention of students with hyperactivity. When seen as a text genre, Barbe [14] has shown that word-based puzzles have been used for political propaganda and as "instrument for powerful manipulation".

2.1.3. Patents

Seeing prices of $110 to $260 [51] encourages to keep economic goals in mind while dealing with puzzles. The financial interest can also be concluded by seeing that people invested in filing patents on their puzzle inventions. The first patent appeared in 1936 by Robert [145]. These patents usually focus on searching words that need to be filled into a grid. They are thus called "word search puzzles", for example, by Jr [94], Chan [29], Seaberg [153], Schroeder [152], and Nichols [125], but they always have some sort of variation to the classic crossword puzzle by providing a particular grid and often give a list of clues. Leonard Joseph Kemp [103] does call his invention an "Educational Puzzle" but it is not different from a regular grid puzzle. Some patents are issued to the combination of different puzzle types like Breeler [25] who combined an integrated crossword and circle-a-word puzzle, or Graybill [79] who mixed word-search, word-link, trivia and word-scramble puzzles. Nevertheless, also utterly new puzzle types like the three-dimensional word-search puzzles by Ditter [59] were issued a patent. Only a few patents focused on electronic devices to play puzzles like Walker and Jorasch [174] and Ghaly [76] and only two patents focus on the generation of puzzles with backtracking: Rehm [140] and Rehm and Rehm [141].

2.1.4. Software

Some software to create puzzles exists. None of these explain their algorithm, though, and the creation process is not transparent or influenceable. These softwares only create American style puzzles, grid puzzles and word search puzzles (without solution words). None of

them create Swedish style puzzles with fully connected grids or word search puzzles with solution words. Being available as desktop software or as web app without an API, they are not easy to integrate into existing micro-architectures. The full set of requirements, described in Section 3.2, is not met by any of these proprietary solutions.

- *1-2-3 Word Search Maker* [1]
- *Armored Penguin* [10]
- *CMFC APP* [36]
- *CPT Crosswords* [39]
- *CrossFire* [40]
- *Crossword Compiler* [42]
- *Crossword Express* [43]
- *Crossword Express Pro* [44]
- *Crossword Hobbyist* [45]
- *Crossword Labs* [46]
- *Crossword Solver* [47]
- *Crossword Weaver* [48]
- *Discovery Education's Puzzlemaker!* [58]
- *Education.Com Crossword Puzzle Maker* [62]
- *My Worksheet Maker* [124]
- *Phil* [132]
- *Puzzle Maker* [138]
- *Suchsel-Generator* [167]
- *Die Suchsel-Maschine* [57]
- *Teacher's Corner Crossword Puzzle Maker* [171]
- *Wolfram Demonstrations Project* [180]
- *Word Search Generator* [181]
- *Word Search Maker | Education.Com* [183]
- *Word Search Maker* [182]
- *XWords* [184]

2.2. State of the Art

"I think, in history, everything is about the remix."

- Jonathan Anderson

As Keiran King [97] states "The New York Times crossword, first published in 1942, is the gold standard of American puzzles, attempted by an estimated 500,000 solvers daily." Looking into existing research mainly work on American-style crossword puzzles and Sudoku puzzles can be found.

2.2.1. Creating Puzzles

Smith and Steen [160] describe a system which compiles partial crossword puzzles by selecting words from a dictionary. Meehan and Gray [113] compare a word-by-word approach using different heuristics in placing words and a constraint approach with a word-by-word and a letter-by-letter basis for filling dense blank grids with letters to create valid crossword puzzles. Mazlack [111] first failed at placing whole words and found a letter-by-letter basis to be successful. Ginsberg et al. [78] researched some puzzle construction mechanisms for American style crosswords. They conclude that lookahead and cheapest-first heuristics are necessary in some form. Also, it is more efficient to use exact information available during runtime instead of using statistical information during compile time, and that the connectivity heuristic is best used while backtracking. Harris, Forster, and Rankin [80] introduced the concept of basic blocks for unconstrained crossword puzzle solutions, *the Crozzle*, and an algorithm to generate the complete set of basic blocks from a lexicon. They observed a correlation between the number of basic blocks present in a lexicon and the occurrence of basic blocks in a humane solution. They also gave an overview of the up to date research on word games [81] and introduced a backtracking algorithm called the ripple effect in 1994. Smith [159] created a suite of programs called XENO to assist in the creation of crossword puzzles. Emrah Aydemir and Zulfu Genc [64] created an online dynamic cross-puzzle generation algorithm for Turkish puzzles.

While the positive effects of puzzles in education have been shown, some researchers focused on specifically creating puzzles for teaching. One of the four educational tools developed by Bailey, Hsu, and DiCarlo [13] is, in fact, a basic crossword puzzle. Arora and Kumar [11] used the motivation to use crossword as teaching aids and thus extend the creation of crossword puzzles in a tool called *SEEKH*. Astutik [12] focused her study "on implementation of Word Search Game to develop students' mastery of vocabulary".

Genetic Algorithms

Bonomo, Lauf, and Yampolskiy [23] describe generating crossword puzzles as NP-hard problems that are candidates for genetic algorithms (GA) which they combine with a Wisdom of Artificial Crowds (WoAC). They focused on American crossword puzzles. Kyle Williams [100] used a genetic algorithm and shows that the values for the variables have a significant effect on the performance of his genetic algorithm. The algorithm converges quickly to a sub-optimal solution but finds an optimal solution over time through mutations. Engel et al. [65] use a two-step process to generate Swedish crossword puzzles. In the first step, they use a memetic algorithm to create puzzle masks, and in the second step, they fill the mask with words from a given dictionary. They also show that this process is NP-complete.

Steepest Ascent Hill Climbing Algorithm

Sazaki et al. [149] use the *steepest ascent hill climbing* algorithm to create crossword puzzle boards. They also prove that the larger the grid, the fewer black boxes are in the board. Then, in [148], they also compare it to a genetic algorithm concluding, that the fitness value of the genetic algorithm has a higher value. However, the boards formed by the steepest ascend hill climbing algorithm are more optimal.

CSP Algorithms

Beacham et al. [15] used *constraint satisfaction programming* to generate crossword puzzles. Tomozov [172] combines constraint satisfaction with simulated annealing for his crossword construction. Wilson [177] has used an integer programming approach for crossword compilation which he then discarded in favour of a more straightforward approach. He concludes that while integer programming is a good alternative for solving logical problems, it showed not to be successful in crossword compilation. Botea [24] uses a hierarchical CSP encoding with word slots as variables at the high level and grid cells as variables for letters on the low level. Samaras and Stergiou [147] use crossword puzzle generation as a structured problem to empirically compare the performance of non-binary and binary constraint satisfaction problems.

Predicate Logic

Berghel [17] shows a way of expressing the puzzle generation problem as first-order predicate logic. Berghel and Yi [19] also complement their earlier results on crossword compilation by describing a crossword compiler-compilation.

2.2.2. Vocabulary Creation

Rigutini et al. [143] present a system that automatically generates crosswords, with web extraction and natural language processing for definitions. Later they presented a crossword generator [144] that crawls the web to extract a vocabulary with associated definitions using natural language processing and compiles a crossword using constraint satisfaction programming. Ranaivo-Malancon et al. [139] present a clue answer building algorithm to be used as a separate component in automatically building crossword. They rate 53% of the fill-in clues generated by their algorithm to be correct. Aherne

and Vogel [3, 4] use WordNet as a lexical source to generate "themed" crosswords. Ali, Black, and Spiro [7] investigate the efficiency of using databases or text files as word banks for generating crossword puzzles. Pintér et al. [134] developed a general automated word puzzle generation method from unstructured documents, from a topic model and semantic similarity measure of word pairs. Espinosa-Anke et al. [67] introduce DefExt, a semi-supervised Definition Extraction Tool that uses text corpi to learn terms and their core features and definitions. Esteche et al. [68] introduce a similar system that uses Spanish texts to create crossword puzzle boards and definitions. They use pattern matching for definition extraction and a greedy strategy for the crossword generation. Smith and du Boulay [157] describe a program to generate certain classes of cryptic clues that can be used in conjunction with other crossword compiling software.

2.2.3. Solving Puzzles

Littman, Keim, and Shazeer [107] solve crossword puzzles by giving a set of clues and a crossword grid. Their probabilistic cruciverbalist *PROVERB* separates the problem into two steps, candidate generation and grid filling. First, they find all candidate answers to clues from a knowledge base. These candidate words are then fit into the puzzle grid. In their second attempt at *PROVERB* Littman, Keim, and Shazeer [106] and Keim et al. [96] split the program architecture into expert modules that generate the candidate words from different sources. The lists of candidate words are merged before the grid filling is executed. Ernandes, Angelini, and Gori [66] introduce WebCrow, a system to solve crossword puzzles by searching the web without having ledge or a database. It utilises machine learning, information retrievle as well as natural language processing and is thus designed to support multiple languages. Pinter et al. [133] introduce an automated word puzzle generation via topic dictionaries. The puzzles generated are the *odd one out*, *choose the related word*, and *separate the topics*. Manzini, Ellis, and Hendler [109] developed a pipeline for IBM's Watson to solve clues of word puzzles. Their system has been

developed to solve syllacrostics, but also discuss how the approach can be used for other word-based puzzle types.

2.2.4. Other

Not all work is focused on creating or solving puzzles. C and Thorpe [28] introduce a methodology to count the number of crosswords possible in a language. December and Andreasen [55] show that there is a strong connection between the complexity of a language and the possibility to create crosswords which directly leads to entropy the most critical measure in information theory. Pershits and Stansifer [131] compared two strategies to build crossword puzzles. One approach systematically tries the possibilities, throwing away partially completed puzzles, the other builds up the diagram word by word. Efron [63] modelled crossword puzzle difficulty in a probabilistic way, where the puzzle difficulty P is dependent on the probability of solving a clue/answer set that has a probability of $P(A|C)$ based on the probability of the answer and clue appearing together in a language. While the model performed well, some issues appeared that were not addressed, for example, that solving all horizontal words, reveals the answers for vertical words without needing to solve them manually. In future work, the question of "How much information is in a particular puzzle?" should be examined.

2.2.5. Other Puzzle Types

jigsaw

Toyama et al. [173] use a genetic algorithm to solve jigsaw puzzles with rectangular fields by looking at the pixel values on the borders of the pieces. Sugiyama, Osawa, and Hong [168] define two abstract puzzle models, permutation puzzles like Rubik's cubes and cyclic puzzles like the Rubik's clock, which can be modelled as a graph.

They then introduce puzzle generators and use graph drawing as a method to layout the puzzles.

Word Search

Stanovich and West [165] found that the surroundings of searched words influence the speed at which terms are found. Order of speed, increasing: non-word fields, pseudoword fields, word fields.

Sudoku

Sudokus are among the most popular puzzles. Its numeric nature and the NP-completeness rose the interest of researchers while not only generating and solving is interesting, but Sudokus can also be used as benchmark tests for newly created algorithms. Quite often, genetic algorithms have been used, for example, by Milton and Ortega-Sanchez [118] to generate sudoku puzzles modelled as a constraint optimisation problem. Contrary to that, Dr. John M. Weiss [60] shows that while genetic algorithms usually are effecting in attacking NP-complete problems, they are quite ineffective in solving Sudokus because of slow convergence, inability to escape from local minima and a variety of other problems. He concludes that solving Sudoku puzzles is a GA-hard problem. More focusing on algorithmic details, Cox, Lucci, and Pay [38] show which effects dynamic variable and value ordering heuristics have on searching for Sudokus.

A huministic approach to solve Sudokus, which does not guarantee a result but achieves an overall good time combined with a slow brute force approach, which ensures a solution, is presented by Sruthi Sankar [164]. Genetic algorithms also found their way to solving Sudokus [161] and multistage genetic algorithms have been used so solve hard Sudokus, that have fewer clues but require a unique solution [30]. *Solving Every Sudoku Puzzle* [162] uses two ideas to solve a Sudoku: constraint propagation and search. Christensen and Emerson [34] use CSP instead of brute-forcing to search for sudoku

solutions in an efficient way that is close to the intuitive solution methods used by humans. Rohit Iyer, Amrish Jhaveri, and Krutika Parab [146] review Sudoku solving with the following patterns: The Naked Singles Pattern, The Hidden Singles Pattern, The Locked Candidates Pattern, and The Naked and Hidden Pairs Pattern. They conclude that using patterns lowers the execution time when solving a massive number of Sudokus. Murphree [123] solve Sudoku as a constraint satisfaction problem and compare the heuristics Backtracking with forward checking and minimum remaining value, arc consistency and arc consistency pre-processing. Lynce and Ouaknine [108] code the Sudoku problem in conjunctive normal form to solve the puzzle using polynomial-time satisfiability inference techniques. Their inference method does not perform well or not at all on very hard puzzles which would require more sophisticated inference techniques. Hughes and Yampolskiy [88] describe a genetic algorithm with a wisdom of crowds heuristic to solve sudoku puzzles.

Yong-zhuo [185] defines how to decide on the difficulty of Sudoku puzzles by looking at four aspects: "total given cells, distribution of given cells, applicable techniques of logic deduction and complexity of enumerating search." They then create puzzles in a two-step process: "to create a valid grid by Las Vegas algorithm, and then to generating puzzles by erasing some digits using five operators". Simonis [156] as constraint problems reached the masses a view on Sudokus from a constraint point of view to solve, generate, and measure difficulty. Felgenhauer and Frazer Jarvis* [69] calculates how many sudoku puzzles can be created and come to the result of $\approx 10^{21}$. However, he already knows that this number cannot be correct as some factors need to be removed due to relabelling, which leaves $2^7 * 27704267971$. A year before Felgenhauer and Jarvis [70] came to the same result while starting from knowing that $\approx 10^{27}$ latin square puzzles exist. Zambon [187] another calculation on how many sudokus exist and can be created. Moraglio, Togelius, and Lucas [121] introduce geometric crossover with the example of sudoku puzzles. Milton and Ortega-Sanchez [118] discuss the design and analysis of a configurable genetic algorithm. They use for solving sudoku puzzles as test-bed since it is a constrained optimisation problem belonging to the NP-

complete class of computational problems.

Others

De Kegel and Haahr [54] researched procedural content generation, such as puzzles. Smith et al. [158] created a puzzle type, the Cross-Song, which combines the pattern-learning and pattern-seeking in music and puzzles. Port and Yampolskiy [135] created a hybrid algorithm that is made up of a genetic algorithm and the wisdom of artificial crowds effect to improve the GA results. It is used to solve solitaire battleship puzzles. Ntoa et al. [127] presented A-Cross, an accessible crossword game for blind and visually impaired users. While still having some future work todo, the implemented prototype already enhances usability and accessibility by suggesting the next clue to be answered. This is done by selecting a clue that is related to the current word and has the fewest missing letters. Purdin and Harris [137] attempt to solve the go-words crossword variant, known to be NP-complete, using a genetic algorithm. The New Yorker Henriquez and Maynes-Aminzade [83] introduces a cryptic crossword puzzle where the clues are cryptic. These cryptic clues are made of two parts, the definition (straight) and a wordplay (cryptic) element.

2.2.6. Benchmarks

Berghel and Rankin [18] provide a dataset for measuring the relative efficiency of different crossword compilation methods which they later refined [163].

3. Problem Statement

"It always seems impossible until it's done."

- Nelson Mandela

In this chapter, the field, this thesis is trying to research, is described. It is split into use cases, specifying who will use the created algorithms, as well as how it is going to be used. Additionally, the requirements for a high-quality puzzle as well as the goals for this thesis, are stated.

3.1. Use Case

"Knowing is not enough; we must apply. Willing is not enough; we must do."

- Johann Wolfgang von Goethe

How should the user interact with the software? What is exposed, and how can the puzzle compilation process be started?

The editorial staff creating a puzzle should be able to use the software without thinking about the algorithm. The workflow consists of selecting or creating a vocabulary, which might be themed, setting a solution word, choosing the dimensions of the puzzle, and marking restricted areas, that are left blank for images, advertisements, or text. This puzzle data is then submitted to the algorithm which

compiles the data into a puzzle grid. After assembling the puzzle, automatic quality assurance tests (QA), as well as manual quality control, are done to check the puzzle and feasibility of publication. The process of preparing the data and manual quality control is done in a graphical user interface, also called the front-end, which is not part of this work. The focus of this thesis is the algorithm itself that accepts the raw data and compiles it into a playable puzzle grid. To easily integrate the algorithm into an existing software stack within a micro-service architecture, already consisting of the front-end, a REST API is available to interact with the algorithm. The data is sent to the algorithm with an HTTP POST request containing the data to start a compilation process. The connection is held open until the puzzle compilation process is finished and the HTTP response includes the compiled puzzle or error codes.

In later steps, it makes sense to introduce a job queue so that the HTTP connection can be closed earlier during long-running compilation processes. The initial request would then contain a callback address, which is called after the compilation process, and to which the created puzzle is sent. This is not part of the requirements of this thesis.

3.2. Requirements

> *"Gamers are some of the toughest people to please. They have extreme requirements. They want everything."*
>
> - Lisa Su

There are multiple levels of requirements concerning the process of creating puzzles, containing quality measurements for vocabularies, puzzles, and the algorithm itself. This section introduces these specifications. Even if the necessity of some requirements is not immediately apparent or not directly relevant to the developed algorithm, they are listed here to understand the whole picture and the decisions

made. While the extensive set of requirements is listed in full detail in Chapter A, this section only covers the relevant requirements for this work and focuses on word search puzzles with solution words as described in Section 1.2. Some requirements are weak or "nice to have" and do not count into the feasibility study of the examined technologies but will be covered in future work.

The following requirements are not consecutively numbered, are never re-numbered, and are not arranged hierarchically as this might conflict with already used references[1]. To reference requirements the scheme used is RQ.G.01 where the numbers are incrementing over time.

Dechter and Meiri [56] state that experiments using crossword puzzles for constraint programming "reveal the crucial parameters of a problem domain". As with all agile processes, the requirements change over time when gaining new insights, during development and while testing intermediate results, but also due to evolving preconditions. This section describes the latest state and does not necessarily reflect the initial set of requirements.

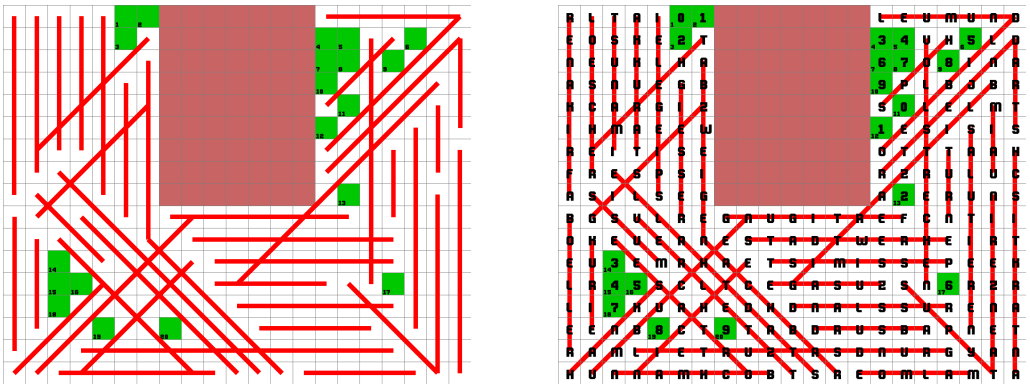
When describing fields in images, (x,y)-coordinates are used. They are in the fourth quadrant in a two-dimensional cartesian coordinate system with the origin in the top left, starting to index fields from 0. For example, (4, 3) describes field five from the left and four from the top.

3.2.1. Requirements for Grids

RQ.G.00 Word Directions

Words can be forward in a top-to-bottom and left-to-right manner and backwards in a bottom-to-top and right-to-left manner as well as diagonal.

[1] http://www.smartmatix.com/Resources/RQMTipsTraps/NumberingTraps.aspx (Accessed on: 2020-05-24)



(a) Empty Grid (b) Filled Grid

Figure 3.1.: A grid with restricted fields (dark red fields) and solution word (green fields), with empty positions on the left (red lines) and filled with words and solution word on the right.

RQ.G.01 Line Patterns

A line pattern, as seen in Figure 3.1a, describes the positions where words can be placed without considering the words' directions. That means each position has two field coordinates that connect a line where a word can be placed. A grid extends a line pattern by explicitly specifying word directions by stating which field is the start and which is the end of the word.

RQ.G.02 Intersection

Words can overlap each other as seen in Figure 3.1a and Figure 3.1a, for example, in field (4, 3).

- Two words may overlap with a maximum of one letter.
- Each word may be overlapped in a maximum of one-third of its letters (for example a word with six letters may only overlap two times).
- Words can not contain other words.

- "Physiotherapist" is only one word, "IOT", "other", "the", and "rapist" would not be allowed as words.
- "Startup" excludes the words "start" and "art".

RQ.G.03 Word Length Variability

The grids need alternating word lengths because a puzzle is not very interesting and fun to play if only 2,3,4 letter words are used.

RQ.G.04 Solution Word

When the player has found all words:

- Some fields might be leftover and contain random letters to create a letter desert puzzle.
- If the puzzle has a solution word the leftover fields spell the solution word from top left to bottom right
- If the solution word length cannot be met, a puzzle with a solution word length deviation of up to 50% can be returned. The solution word can then be manually adapted.

The solution fields in Figure 3.1a are filled with the solution word "01234567890123456789" in Figure 3.1b.

RQ.G.05 Restricted Fields

The puzzles might leave out some fields to place text or ads into the puzzle. The editor specifies these restricted fields. An example can be seen in Figure 3.1a, where the restricted area is coloured in red.

RQ.G.06 Even Orientation Distribution

The distribution of word orientations needs to be well-balanced, preferably in a uniform distribution. Since the uniform distribution was hard to achieve this requirement was adapted, that each orientation must occur at least 10% of the time. Words can go in all directions:

- Left-to-right and right-to-left (HORIZONTAL).
- Top-to-bottom and bottom-to-top (VERTICAL).
- Top-left-to-bottom-right and bottom-right-to-top-left (DIAGONAL).
- Top-right-to-bottom-left and bottom-left-to-top-right (ANTIDIAGONAL).
- It should be possible to dis-/enable directions.

RQ.G.07 Maximum Field Population

In each field, a maximum of two words is allowed to overlap each other. For example, if two words overlap in one field, the population counter is two if three positions overlap in one field, the population of that field is three.

RQ.G.08 Shaped Puzzles

By providing fields that restrict the outside shape of the grid, specially shaped puzzles should be possible.

3.2.2. Requirements for Puzzles

RQ.P.01 No New Words

Words next to each other must not create compound words together. For example "basket", "ball", and "basketball" are three valid words,

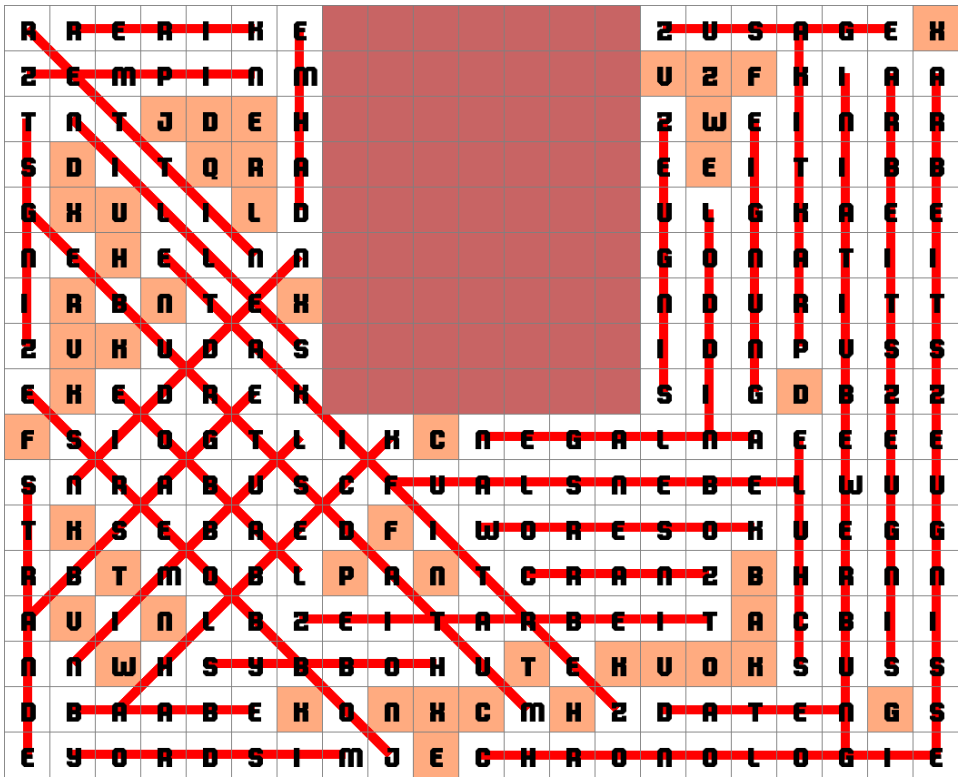


Figure 3.2.: A letter desert puzzle with restricted fields (dark red fields) which were given and needed to be left out by the algorithm.

but "basket" and "ball" might not be placed in one row next to each other.

RQ.P.02 No Random Words

Words should not be created by coincidence by a lucky placement of the letters on the grid. An example can be seen in Figure 3.2 where the horizontal word "zwei" starting in the field (14, 2) is not listed in the list of words to search for and in this case, it was not even contained in the given vocabulary.

3.2.3. Requirements for Vocabularies

RQ.V.01 Vocabulary Priority

The puzzles can be created from two or more vocabularies (lists of words)

- A primarily used themed vocabulary (e.g. medical puzzle, soccer, Christmas)
- A backup vocabulary to fill in blanks when the themed vocabulary is not enough to create a puzzle
- Multiple different puzzles should be possible to generate from the same two vocabularies
- Only one vocabulary is required, if both are provided one has a higher priority for use

Some words need to be prioritized

- Themed vocabulary that needs to be prioritized, the rest of the puzzle is filled with words from a second vocabulary
- Prioritized vocabulary with for example 5 words that have to be used

Example: a puzzle with a medical focus for an Austrian customer, who provided a few customer-specific terms

1. Customer-specific words have to be used
2. "Theme medical" words should be preferred
3. "Theme German Austrian" words should be used randomly at third-highest priority
4. "Theme German general" words should be used randomly

RQ.V.02 Randomized Words

If, for example, five puzzles are created in one run, they should not contain the same words. This could be achieved by dynamically shrinking the vocabulary on interim solutions or by creating blacklists.

RQ.V.03 Blacklists

Some words need to exclude each other.

- For example, if a German chancellor is used in the puzzle, no other chancellor should be used.
- Example for a softer requirement: if the puzzle is big enough, two German chancellors might be used, but they need to be placed far away from each other.
- Could be implemented as a constraint or as QA to discard puzzles that do not meet requirements.

Blacklists prevent words from being used together

- Blacklists have high priority as they directly affect the quality of the puzzles.
- They prevent the usage of near-duplicates (synonyms, thematic similarities, and others).
- If no puzzle can be generated, exclusion criteria can be disabled to give the algorithm more options.
- Words must not appear twice in a puzzle.
- Small puzzles must not have near-duplicates at all.
- Larger puzzles might have near-duplicates but not in the same corner.
- Words can also exclude clues. For example, if a clue "Name of North America" with the answer "USA" is used, other clues including "USA", for example, "State in the USA" should be excluded.
- Worst case: two crossing words where one clue holds the answer to the other word.
- Opa and Mutti should not be in the same puzzle (soft requirements/nice to have) (e.g. word embeddings)
- Mama and Mutti must not be in the same puzzle (hard requirement/exclusion) (e.g. synonyms)
- Prevent conceptual aggregations in a corner of a puzzle (e.g. Italian poet, Italian city, Italian river might create an Italy themed puzzle by accident).

While blacklists prevent from using similar words together in one general puzzle, this effect might be at a disadvantage for themed puzzles. By providing a themed vocabulary, a themed puzzle should be possible. While global, general blacklists would prevent for example the words "Rome", "Italy", and "Latin" to be used together in one puzzle, they would be necessary to be used in combination to create an Italy themed puzzle. Another example, most monkey names end on "-affe" in German, so these words usually exclude each other because they are too similar. However, a monkey themed puzzle might contain multiple monkey names. This might need disabling parts of blacklists or the use of multiple blacklists. For example, a word-in-word blacklist should always be active, while a blacklist based on words similarity or semantic closeness could be disadvantageous for themed puzzles.

RQ.V.04 Vocabulary Size

Vocabularies of differing sizes need to be usable. Vocabulary sizes can range from small themed vocabularies with ~30 words to full vocabularies with ~70,000 terms. While the small vocabularies should produce usable results, the large vocabularies should still generate results in a few minutes.

RQ.V.05 Words Only Once

Each word may only be used once per puzzle.

3.2.4. Requirements for Puzzle Algorithm

RQ.A.01 Quality

Instead of returning no result, the algorithm should return the best it can achieve for manual completion. Accepted loss of quality:

- Not matching the desired number of solution letters.
- Low overlap factor.
- Not including a solution word at all.
- Returning a boring line pattern.
- No uniform distribution of orientations.

RQ.A.02 Pre Assignment

It should be possible to pre-populate letters and words to positions and fields.

- Preassign a puzzle (grid) with some letters
- Preassign a puzzle (grid) with some words

RG.A.03 Output

The algorithms returns:

- A list/array of fields containing the letters.
- A list of words used.
- The solution.

RG.A.04 Input

The generated puzzle should be configurable:

- Width in number of fields in integer.
- Height in number of fields in integer.
- Which areas should not be used (to place description text or advertisements), as a list of restricted fields.
- Which directions for words are allowed as boolean switches.
- The vocabularies that should be used, as comma-separated strings.
- The number of words to be used in letter desert puzzles as integer.

- If fields are permitted to be left out.
- If a solution word is needed.
  - And if, the word itself as a string.

RG.A.05 Maximum Runtime

The algorithm needs to stop if no puzzle was successfully created within 30 minutes. It will run on an Intel Xeon E5-2603 with 1,7 GHz and four cores in a virtual machine. The timing needs to be evaluated on that machine.

3.3. Expectations

> *"Setting goals is the first step in turning the invisible into the visible."*
>
> - Tony Robbins

The goal of this thesis is to evaluate if word search puzzles can be generated with constraint satisfaction programming or with neural networks and to develop a functional and usable (see 3.1) algorithm to create puzzles. While it serves as a feasibility study for multiple puzzle types, the focus is set on generating only one type, word search puzzles, while using Swedish puzzles to evaluate the transfer of chosen methods to other puzzle types. While starting with fundamental requirements, only a few features and low (artificial) intelligence the algorithms created are to improve iteratively becoming smarter, more versatile, and produce higher quality results. Furthermore, methods of artificial intelligence are evaluated for usability. By-products of the evaluation are strengths and weaknesses of the investigated methods as well as insights into the difficulty of puzzle generation.

Finally, a functional algorithm is presented to be used by puzzle agencies. Each algorithm is running individually as a web service

that is accessible via an API (REST or GraphQL) to be integrated as separate services in a microservice architecture. As mentioned above, the focus is not set on handling or maintaining vocabularies but on the creation process of puzzles only. In the end, editorial staff should be supported in the puzzle process by allowing the editor to focus on editorial work like curating themed vocabularies instead of manually placing words in a grid.

4. Method

"Take a method and try it. If it fails, admit it frankly, and try another. But by all means, try something."

- Franklin D. Roosevelt

In this chapter, the two evaluated methods, constraint satisfaction programming (CSP) and deep learning (DL) with neural networks (NN), and the procedures are explained. Due to the original idea to compare these two methods, the chapter is divided into two parts. As the method in detail and the evaluation in chapter 5 will show, this division has no advantages in the future but reflects the procedure of these investigations.

4.1. Constraint Satisfaction Programming

"In nonfiction, you have that limitation, that constraint, of telling the truth."'

- Peter Matthiessen

Before going into the implementation, some definitions need to be made, to understand the functionality of the algorithms

4.1.1. General Data Structure

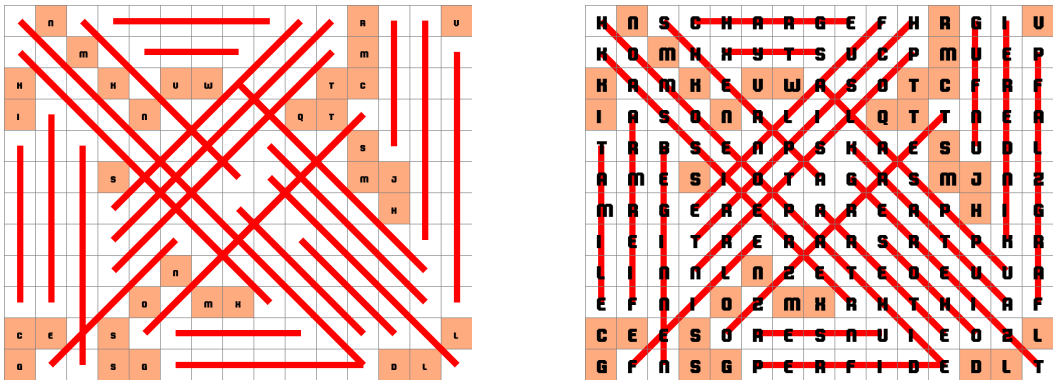
The presented data structure abstracts the information from the puzzles. Ideally, the structure does not know about the puzzle type and thus can be used for different puzzle types with minimal adaptation. The presented grid-based puzzles can be seen as a matrix with origin in the top left corner with one letter per field and words that can cross in these fields. The general abstraction is kept universal, but the constraint problem is implemented with Google OR-Tools by Perron and Furnon [130], so some definitions are specific to the implementation.

Grid

A puzzle grid is a container holding multiple positions arranged in a line pattern. These positions specify the orientation, length and direction of the words to be placed. The Grid object can be created in multiple ways, either by parsing an XML file, as the output of the GridGenerator algorithm or as the result of the GridExtractor algorithm. Positions in the grid are extended with OR-Tools variables which are used while trying to find words that fit into the puzzle. An example of an empty grid can be seen in Figure 4.1a.

Puzzle

The result of the GridFiller algorithm, a filled grid representing Word Search and Swedish puzzles. After finding a solution, the GridFiller returns the puzzle created from the given vocabulary and a puzzle grid. The empty grid seen in Figure 4.1a with filled word positions to create a puzzle can be seen in Figure 4.1b.



(a) Empty Grid (b) Filled Grid

Figure 4.1.: A Grid generated by the GridGenerator, before (left) and after filling (right)with the GridFiller. This letter desert puzzle does not contain a solution word.

Field

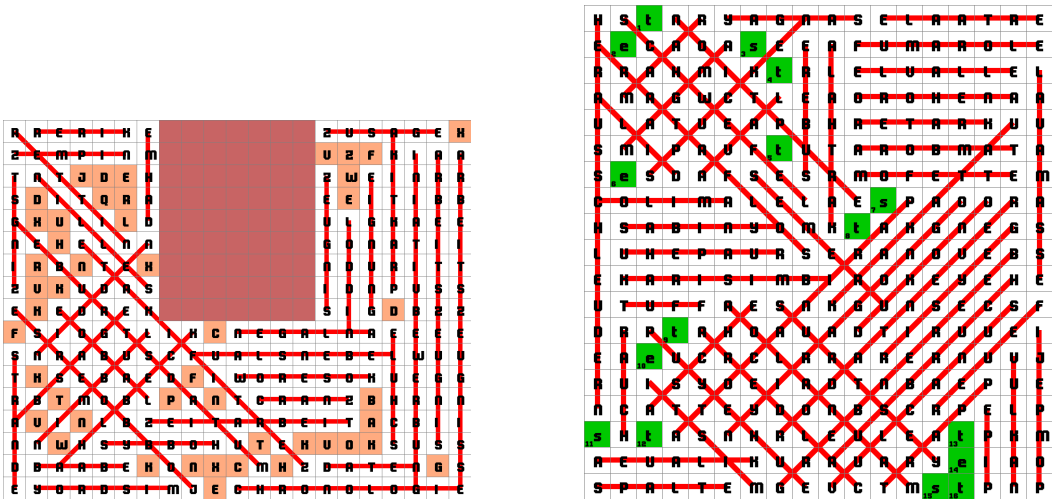
A field on the grid is the container that can be assigned one letter of a word, a solution word letter (word search puzzles), a random letter (letter desert puzzle), a clue (Swedish puzzles), or be empty.

Restricted Field

A field that is not allowed to contain a letter or question. These fields will be left empty by the algorithm, as seen in the red square in Figure 4.2a from (0,7) to (8,13). Restricted fields can be filled with an image, text, or advertisement after the puzzle is generated.

Solution Field

A field containing one single letter of the solution word. These letters can be ordered to read the solution word from top left to bottom right. Alternatively, they can be shuffled to create another puzzle where the randomised letters have to be ordered to find the solution word. An



(a) Puzzle with restricted fields

(b) Puzzle with solution word

Figure 4.2.: Two Puzzles, one with Restricted Fields (dark red fields) on the left and one with Solution Word on the right (green fields).

example can be seen in Figure 4.2b field (2,0) for the solution word "testtesttesttest".

Empty Field

A field that contains a random letter that is not part of a position/word or a solution word. These are empty as they are not filled by the algorithm, but by another step where random letters are distributed. These empty fields are only relevant for letter desert puzzles without a solution word. The orange fields in Figure 4.2a are an example of randomly filled empty fields.

Question Field

A field containing a question to serve as a clue while filling a Swedish puzzle. Each question field is connected to a position. An arrow links these question fields with their corresponding answer positions

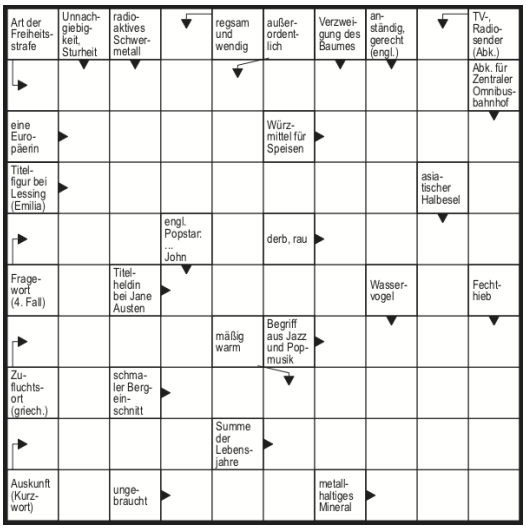


Figure 4.3.: Swedish puzzle with question fields and arrows pointing to the position the Questions answer belongs to.

indicating where to place the answer to the question. An example can be seen in Figure 4.3.

Position

A position is a container in a grid that can hold a word. It has start and end coordinates, an orientation, and a length. For example, the grid position (0, 3) to (0, 8) in Figure 4.1a has a horizontal orientation and a length of six letters.

Orientation

Orientation describes the directionality of a position. Four base orientations are available, that, when rotated by $180°$ make a total of eight possible orientations. The four base orientations are:

- HORIZONTAL
- VERTICAL

- DIAGONAL
- ANTIDIAGONAL

Solution Word

A solution word is made up of a term and a list of solution fields that can be assigned the individual letters. The GridGenerator algorithm is passed a solution word and places positions for the solution word letters on the grid. The algorithm then returns these fields for the GridFiller to insert the solution word.

Word

Vocabulary item, consisting of a string of letters. It is assigned a UID that can be assigned to a vacant position by the GridFiller algorithm.

Vocabulary

Contains a list of words. These words can be shuffled to be fed into the fill algorithm in random order preventing a deterministic result and creating a new puzzle on every run even if the grid and vocabularies given are the same.

Vocabularies

A list of multiple vocabularies that are fed to the GridFiller algorithm. Even if the individual vocabularies are shuffled, these lists keep the order to maintain the priority of the given vocabularies.

Blacklist

Holds information about words excluding each other as described in requirement 3.2.3.

Distribution

A function to sample position lengths when the GridFiller creates a new position. While the samples are drawn from a normal distribution, the length can be limited by giving max and min lengths that can be derived from the width and height of the grid. Usually, this distribution is initialised with mu, sigma from a given vocabulary and size of the grid. The grid size limits the word lengths.

Vocab Distribution

A different approach to distribution, sampling from the probability distributions of word lengths based on a given vocabulary. While the simple distribution can create lengths that are not contained in the vocabulary, this distribution is limited to available lengths with higher probabilities for word lengths that appear more often in the given vocabulary.

Vocab Lengths Generator Function

The third approach is replacing the distribution function. While it is technically not a distribution, it has a sample function to act as a drop-in replacement for the other two distributions. This "distribution" keeps track of the available word lengths. Once a length is used in the grid, it is removed from the list. This way, a precise set of position lengths perfectly matching a given vocabulary can be created. If the list is empty and the grid needs more, it automatically resets to its initial state. The distribution has three sample modes: random lengths, decreasing order, increasing order. In the first mode, the

lengths are randomly sampled, giving an even distribution. In the latter two modes, the lengths are sampled by decreasing or increasing order. For example, with descending order, long position lengths are sampled first before returning shorter lengths.

4.1.2. Initial Intuition

The initial implementation considered each word as a variable and calculated each possible placement on the grid as the domain for the variables, calculating allowed overlaps to create "allowed assignments". This approach did produce acceptable results for small puzzles but showed to be very inefficient for large vocabularies and large grids as the number of intermediate variables grows exponentially. This prototype had to sample the vocabulary to limit the number of intermediate variables which limited the algorithms possible word arrangements leading to multiple runs with newly sampled vocabularies before being able to generate a puzzle. As this sampling was not suitable, and the need for large vocabularies (Requirement 3.2.3) was given, this idea was not pursued further. The following approach is based on the architecture of splitting grid generation (or line patterns) from filling these grids. By defining a grid data model, a GridExtractor algorithm is introduced to extract the grid from existing, handmade puzzles. This way it possible to develop a GridFiller algorithm before/while developing the GridGenerator separately.

4.1.3. Generating Grids

The GridGenerator algorithm is split into three phases, that can individually be enabled: placing new positions, extending positions to adopt empty fields, and covering up empty fields. Additionally, different quality assurance checks are integrated to discard grids that do not meet the requirements immediately.

Phase One: Placing Positions

In step one, the algorithm tries to place as many positions as possible in a backtracking manner by incrementally building on partial solutions. Each partial solution is improved step by step. When in a dead-end, where no final solution is found, and no improvements can be made, the algorithm discards the current intermediate solution and tries to improve the previous intermediate solution with another step.

First, a position length is sampled by the given word length distribution. Then an orientation of the four base orientations and a position from all possible starting fields are randomly selected. When a position was successfully placed on the grid, a new position is tried in the same manner. The decision if a position can be added depends on the overlap counters of possibly intersecting positions determining if a position is not overlapped more than the overlap factor allows (Requirement 3.2.1). For each placed position, a conflict group is calculated, preventing other positions from being placed in the same conflict group. These conflict groups prevent words from containing other words or from overlapping with more than one letter (Requirement 3.2.1). Placing positions on restricted fields is avoided (Requirement 3.2.1). If a position cannot be added, other starting points and orientations are tried in a backtracking style. If no position with the given length can be placed and resizing positions is allowed the position length is shortened by one letter, respecting the minimum word length, starting a new trial. Once no new positions can be placed, phase one is finished.

QA: Orientations Amount

Before going into phase two, a quality assurance check is performed to ensure that enough orientations are available. If not enough orientations have been used, the grid is immediately discarded, and a new trial has to be made. While an even distribution would be desirable, this has rarely been achieved. Therefore, the distribution of positions

is calculated based on the number of occurrences per cent. If at least one running direction falls below the given threshold value, the grid is discarded.

Phase Two: Extend Positions

In phase two, if resizing is allowed, the placed positions are extended to adopt empty fields. Adopting empty fields is done by iterating through all vacant positions and checking if a position ending in a surrounded field can be extended to take up the empty field. This is done as long as the grid contains more empty fields than the solution word has letters (Requirement 3.2.1). Once enough fields have been adopted, or no more positions can be extended, phase two ends. This phase currently does not respect the overlap factor or field population resulting in overpopulated fields or positions in some rare cases.

Phase Three: Cover Up Empty Fields

If phase three is enabled, the algorithm iterates through all columns and rows collecting all empty fields. Then a position with starting and endpoints placed on the first and last empty field in a column/row is created and added. This phase currently does not respect the overlap factor or field population resulting in overpopulated fields or positions with too many intersections and thus discarded grids quite often. It also only places horizontal and vertical positions in the current state.

QA: Max Field Population

Each fields' population count is calculated and compared to contain a maximum number of overlapping positions. While this threshold is dynamic and can be changed, the default setting is set to a maximum of two overlapping positions in one field (Requirement 3.2.1).

QA: Solution Length

The solution length may deviate by a given factor, grids with solution words differing too much are discarded (Requirement 3.2.1). This is a dynamic setting. Matching the targeted solution word length heavily depends on the given vocabulary. By default, the threshold is set to allow a deviation of 50% because an almost finished puzzle, which can be improved manually, is preferred over long waiting times or no results at all.

Flip Position Directions

In the end, positions can be rotated to make puzzles more exciting and challenging. By default, 50% of all positions are turned around to create a grid with alternating directions.

4.1.4. Filling Grids

Filling grids is handled as a constraint problem that is modelled as an assignment problem with Google OR-Tools by Perron and Furnon [130], the CP-SAT Solver is used to solve the optimisation problem as an integer programming problem. Unless mentioned otherwise, the OR tools specific modelling challenges are not mentioned as they are not relevant for the logical answer to the research question.

Initialisation

The algorithm is provided with a grid, containing empty positions, and vocabularies as well as other parameters like shuffling words or using blacklists. If required by the puzzle type, a solution word is provided additionally.

Positions as Variables

Each position is assigned a variable that can contain a word after the solver has successfully finished. Also, each field that contains overlapping words is modelled as a variable that can contain letters. These overlap variables are assigned to all positions that intersect with each other. This way, it can be ensured that assignments of words to a position can only be valid if all overlapping words have a valid assignment with a matching letter in the intersecting field.

Domain

The domain for variable values consists of words identified by unique IDs. It is created by calculating allowed assignment constraints, which take the length of the words into account to allow only words that match the length of the positions. While calculating a possible assignment, it consists of two constraints: the word that can be assigned to the position, as well as the letter that can be assigned to the overlapping field.

Each Word Only Once

To ensure that each word is only used once in a puzzle, an "all different constraint" is used. This is a global constraint that prevents a value from the domain to be used multiple times in the result. This implements requirement 3.2.3.

Blacklists

Blacklists contain words that are not allowed to be used together. While a global constraint like the "all different constraint" excluding the use of two values together would be preferable, this is not possible to be implemented with OR-Tools. So these excluding words are

modelled as forbidden assignment constraints that act as a pair of excluded values for two variables.

Reversing Words

With smaller vocabularies, the algorithm has a hard time placing words. Thus, the directions given by the grid can be ignored. The GridFiller is then allowed to place words for- and backwards. To achieve this, words are reversed and added to the vocabularies before starting the solver. Blacklists to prevent the words from being used in both directions are created automatically.

Decision Strategy

OR-Tools have different decision strategies and is configured to use the following settings. Variables are chosen first; the values are selected by minimum first. This means that words on the top of the vocabulary are tried first during the backtracking phase of the constraint solver.

Randomised Order of Words

As described in the last section, the first words in a row are used first. This means that a copious vocabulary with hundreds of words starting with "A" could result in a puzzle that contains only words starting with that first letter. The Domain is shuffled before running the solver to prevent this behaviour. To keep vocabulary priorities, each vocabulary is shuffled separately.

Prioritised Vocabularies

To establish priority in vocabularies (Requirement 3.2.3), without being able to pass multiple domains, the selection strategy of trying words in a given order is used. This means that words with higher priority need to be further ahead in the list. Thus, if multiple prioritised vocabularies are given, the vocabulary with the highest priority needs to be at the top of the domain. When shuffling, not the whole list may be shuffled, but each vocabulary individually while maintaining the order of the vocabularies.

Complexity

The process starts by iterating all position combinations (O(p^2))) before iterating each pair of possible word combinations O(w^2))) and finally checking if the combination is possible by comparing the possibly overlapping letters of both words. This results in the overall complexity of O($p^2 * w^2$)) for creating allowed assignments. The vocabularies are grouped by word length, to reduce the lookup of words that match the compared positions and improve runtime, which results in a lower runtime than the actual calculated worst-case complexity. Also, only overlapping positions are compared, reducing the complexity further. The method mentioned above of creating forbidden assignments has a much higher complexity. Each excluding assignment has to be added for each pair of variables. This approach has a complexity of O($\#positions^2 * \#words^2 * \#blacklistentries^2$) which unfortunately is quite inefficient for large puzzles with large vocabularies and extensive blacklists.

4.1.5. Word Search Puzzle

A meta-algorithm, called WordSearchAlgorithm, combines the GridGenerator and the GridFiller algorithm to create word search and letter desert puzzles. Different vocabularies have shown to require

differing settings for the sub-algorithms. This meta-algorithm first guesses the best settings and then iteratively tries other settings, while also reducing the strictness of some requirements.

Initialisation

The WordSearchAlgorithm initially calculates the required word length distribution based on the given vocabulary and sets default parameters. The "needed words vocab ratio" is calculated by multiplying the width and height of the grid and subtracting the solution word length. It is then divided by the average word length and by the number of words in the given vocabulary. This ratio is used in deciding for the initial parameters used with the algorithms. It is necessary to do so because differently sized vocabularies need different settings for the GridGenerator and GridFiller. For example, smaller vocabularies need the GridGenerator to use the word length distribution in descending order; otherwise, it cannot create grids. In comparison, this behaves quite differently in larger vocabularies where the word length distribution needs to sample word lengths randomly to not only place long positions in the beginning.

Backtracking Different Settings

The meta-algorithm then iteratively tries different parameter settings in two nested loops. Some vocabularies, mainly smaller ones, do not allow for many overlaps in the positions, while for larger vocabularies more word intersections are preferred. Thus, the overlap factor is dynamically decreased in the inner loop, trying to find a puzzle with the highest overlap factor. For all generated grids, in a dynamic parameter step, the grids are immediately tried to be filled with the GridFiller. When a puzzle was successfully created, the algorithm returns this result immediately. In case no grid could be filled another round with decreased overlap factor is started. Once the overlap factor is down to zero, and still, no puzzle was successfully generated, other parameters are changed in the outer loop, trying with the initial

maximum overlap factor. The whole process is aborted if no puzzle was found within 30 minutes.

4.2. Neural Networks

> *"Deep neural networks are responsible for some of the greatest advances in modern computer science."*
>
> - Jeff Dean

In this section, an approach for generating puzzles with a neural network is examined. Puzzles have variable size, and input vocabularies have different lengths. However, neural networks have fixed input and output dimensions, so for the first evaluation, some parameters are set to fixed values. Also, to prevent unforeseen influence zero-padding, necessary to bring a smaller puzzle to the input dimensions of the neural network, might introduce only puzzles of one size are evaluated. This neural network focuses on word search and letter desert puzzles with a size of 15 by 12 fields and an input vocabulary of a maximum of 100 words. To further reduce complexity, the solution word is left out for initial experiments.

The goal is to train a neural network to learn the structure of puzzles and the arrangements of words and letters, not the words themselves. The semantic meaning of words does not matter either in this case and thus no (pre-trained) embedding layers are used. Another advantage is that it is possible to create puzzles from words that are not known at training time. This is an essential requirement as vocabularies and words are edited and extended regularly to create new and interestingly themed puzzles with words never seen in a puzzle before.

The neural network has been implemented with Tensorflow [2] with the Keras API [33]. For processing data, like splitting data into train and test sets scikit-learn [129] was used. All training was done on Google Colaboratory [22] on a Tesla T4 GPU.

4.2.1. Data Representation

As the Lazy Programmer [102] says "All Data is the Same (in Machine Learning)", one of the main challenges for using neural networks is about finding a representation to feed into the neural network. The most promising representation is to treat a puzzle similar to an image with each puzzle field corresponding to one image pixel.

The Grid can be modelled with numbers, where each number corresponds to one field type. This is not necessary for word search puzzles and is thus left out for now, but would be necessary for other puzzle types. Also, the arrows (~20 variants) showing which clue corresponds to which position on the grid, as seen in Swedish puzzles, is left out for word search puzzles.

The width and height of the grid are the pixel values for neural networks that have to be flattened for dense networks but can directly be used for 2-dimensional convolutional networks. Each field has multiple features, limited to three for word search puzzles: word index, second-word index, and letter index. To represent words and letters as numbers, vocabulary and alphabet indices have to be calculated. To allow the full vocabulary to be modelled, all letters of the basic Latin alphabet are indexed from 0 to 25. To allow for umlauts or other special characters, which are usually not used in crossword puzzles, the alphabet, and the corresponding index would need to be extended with these values before training the network. Each field can then be mapped to a letter of the alphabet and to one or two words that are overlapping the field. To extend the puzzle to allow for more than two-word intersections (excluded by Requirement 3.2.1), more features would need to be added before training the networks. With this numeric representation, the network can be asked: "What is the probability for each value in the features (field type, letter, word) on each puzzle field?"

On the one hand, the vocabulary index needs to be calculated for each puzzle and vocabulary individually to prevent the network from learning specific words but encourage extracting information on the structure of the puzzle depending on the given vocabulary. On the

other hand, the same alphabet representation can be used throughout the whole data set as it does not change from puzzle to puzzle.

4.2.2. Training Data

For training and as a benchmark 424 handmade, high-quality puzzles with dimensions 15x12 fields were available. Since this does not seem to be enough for training a network, the data is augmented as described in Section 4.3.7. Flipping and rotating a puzzle creates seven new puzzles, to keep the dimensions of non-square grids only three new puzzles are possible though. Since quadrupling the training set showed a significant increase in validation accuracy, it was concluded that a more extensive training set is necessary. The CSP algorithm from section 4.1 was then used to generate ~10.000 puzzles without solution words to create a larger data set. With the above method of rotating and flipping and thus four-folding the data set, training was conducted with ~40.000 puzzles. This needs to be kept in mind when comparing the results of both algorithms as the neural network is not expected to perform better than the CSP algorithm if it is trained with a data set created by the latter.

4.2.3. Network Architecture

This section focuses on the best performing architecture, while all evaluated architectures can be seen in chapter B of the appendix.

Initial experiments with generative adversarial networks (GANs) have shown that generating puzzles is quite a complex task. Also, the network input is not an arbitrary latent space made up of random values as seen with most generative networks, but a vocabulary which is called the context space from now on. The approach was thus to find an autoencoder (AE) that can en- and decode puzzles and a second AE that can en- and decode vocabularies. In the following steps, it would then be possible to use the encoder parts of a vocabulary AE and the decoder part of a puzzle AE. These separate en- and decoders

could then be used in a new AE going directly from vocabulary to puzzle or be used in other architectures, for example, GANs or conditional GANs (cGAN) where the vocabulary encoder could be used as the conditional input. Other architectures than AEs and GANs are possible to evaluate as well but are not considered in this thesis.

Optimiser, Learning Rate, Loss Function, and Accuracy

After evaluating several loss functions and optimisers, all following experiments were conducted with the optimiser Adam, with a starting learning rate of 0.001 and binary cross-entropy loss function. Other settings for training chosen are early stopping with the patience of 25 epochs to allow the automatic learning rate adaptation to adequately reduce the learning rate when the validation loss stopped improving for ten epochs. The accuracy is the ratio of correctly predicted values and is calculated by dividing the number of correct predictions by the total number of predictions. The average prediction of a network over the test data gives the accuracy of a trained model. This model accuracy is used to compare the performance of different architectures and models.

Hyperparameters

Some different architectures and hyperparameter settings have been tried and evaluated to find the best hyperparameters. After flattening the data and experimenting with deep neural networks, the focus was quickly set on deep convolutional networks using the 2-dimensional data as described in section 4.2.1 which produced much greater results than the conventional dense networks. The convolution size has shown not to make an enormous difference in results. However, it is important not to introduce any pooling layers as often used in image processing since combining fields with maximum, or average values does remove too much information necessary for puzzle generation. This was experienced when comparing two neural network architectures that only differed in MaxPooling layers, as seen in Figure B.7.

Pooling with standard image data is effective since pixel values of neighbouring pixels often have the same value, and thus not too much information is lost when combining them. However, with the puzzles, where each field contains a separate letter, no information should be left out on while encoding.

Autoencoder for Puzzles

Splitting horizontal and vertical convolutions and training them in parallel branches shows the best results. The architecture for en- and decoding puzzles with the best results is seen in Figure 4.4. This architecture has shown to average its validation accuracy for autoencoding puzzles at around ~99.43%.

Autoencoder for Vocabularies

While starting with the gained insights for autoencoding puzzles, no architecture suitable to autoencode was found. Experiments and the search for a neural network have thus been stopped. All tried architectures (seen in chapter B.3) have shown the best performance after training for one epoch, with decreasing accuracy on further training. With this result, no architecture can be considered to be usable, resulting in no proposed architecture here.

4.3. Preliminary Work / By-Product

> *"Perfection is not attainable, but if we chase perfection we can catch excellence."*
>
> - Vince Lombardi

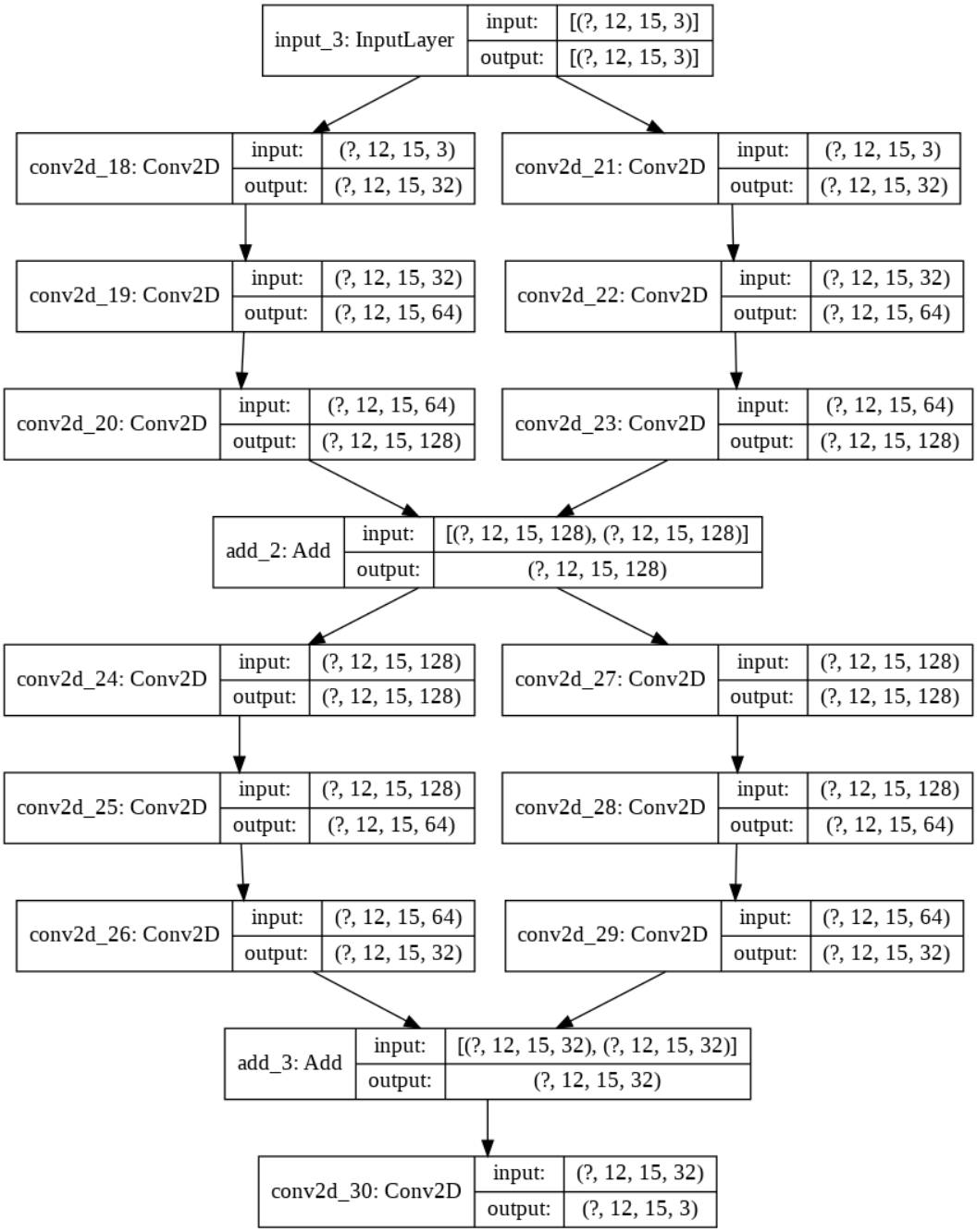


Figure 4.4.: Best performing architecture for a deep convolutional autoencoder with separate horizontal and vertical convolutions that are added in the latent space

Some topics, though not part of the goal of this thesis, were nevertheless taken up and worked on, on the way. These are described in this section. Vocabulary management, for example, is an essential part of the puzzle creation workflow. While some requirements can be fulfilled directly by the algorithm itself, others can be outsourced to the vocabulary management to prepare the vocabularies before they are passed to the algorithm.

4.3.1. Blacklist Creator

A simple word in word blacklist creator was implemented to create blacklists from a given vocabulary.

4.3.2. Synonym Creator

A synonym creator was implemented to fetch synonyms from OpenThesaurus[1] to create a clue list for words used in Swedish puzzles.

4.3.3. Vocabulary Management

An entity-relationship diagram for a possible vocabulary management software was created considering all the different requirements necessary to manage words, clues, and blacklists.

4.3.4. Death Row Analysis

To practice gaining information from available data, a data visualising project analysing death row information from the Texas Department of Criminal Justice[2]. It showed the importance of interpreting the insights correctly. The analysis, for example, revealed that people

[1] https://www.openthesaurus.de/ (Accessed on: 2020-06-29)
[2] https://www.tdcj.texas.gov/death_row/ (Accessed on: 2020-07-20)

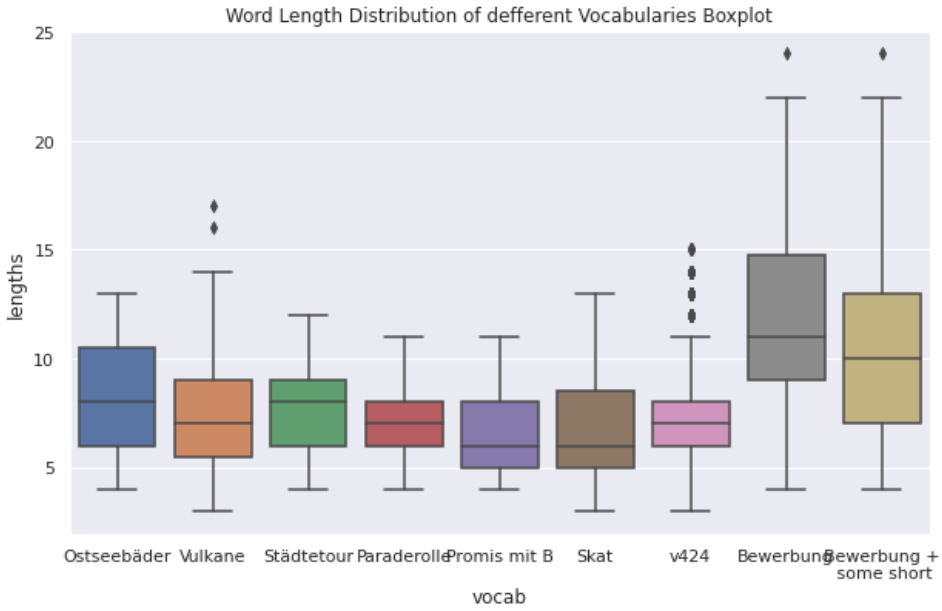


Figure 4.5.: Example for Vocabulary Analysis: Word Length Distribution Box Plot

with the name "Robert" are executed more often in absolute numbers. A conclusion like "I am lucky my name is not Robert" would be dangerous and should not be drawn while ignoring the context. An analysis of all inhabitants of Texas would be necessarily revealing the number of people named "Robert" in the population. Both numbers would need to be put in relation before concluding.

4.3.5. Vocabulary Analysis

A script to analyse and visualise the statistics of vocabularies was implemented to gain insights on the letter and bigram distribution, as well as word length distribution. These statistics were used to improve the settings for the CSP algorithms. The graphs used for evaluation in Section 5.1.1 are created with this script, an example of a word length distribution plot can be seen in Figure 4.5. The visualisations are created with Seaborn [175] and Matplotlib [90].

4.3.6. Horoscope Neural Network

A simple long short-term memory (LSTM) neural network was implemented to train on horoscope data. While this gave insights on using LSTMs, it was concluded that these types of neural networks are probably not suitable for the generation of puzzles. The power of LSTMs is the possibility to process sequences of data, but no sequence was identified in the puzzles. The network hyperparameters were not sufficiently optimised, and the generated horoscope texts could only serve as inspiration for an editor while writing horoscopes.

4.3.7. Data Augmentation by Rotation and Flipping Puzzles

To evaluate how the available puzzle data can be augmented experiments with flipping and rotating puzzles were conducted. Using square puzzles can generate seven new grids from an existing grid, while from rectangular puzzles, only three new grids could be generated. These findings match the results found by Helser [82] who inverts and reverses letter matrices of word search puzzles to create four puzzles out of one.

4.3.8. Timing Experiments

Extensive research has been placed on performance to decrease the runtime of the algorithms. Not surprising it is highly recommended to use Python native functions like list comprehensions, *generator functions* or *Python's itertools* instead of custom (nested) loops. Contrary to expectations, a combination of list comprehensions with the max function is performing better than supplying a key function to the max function though. One of the most valuable insights gained was the use of caching of deterministic values. For example, calculating the length of a word when necessary creates many thousand calculations as the length is needed often. Calculating the length

once during object creation and saving it in singleton word objects for later access was evaluated and found to be useful. Additionally, accessing the stored length of words via direct attribute access proved to be up to 56% faster than using the magic *__len__()* function. Also sorting words by length and storing them in dictionaries proved to be very efficient. As the algorithm often needs words with a length matching the length of a position, lookup performance was increased dramatically by having words accessible sorted by length. The use of *lru_cache* and *cached_property* offered by *Python's functools* does improve performance but is only usable for deterministic, non-changing values. Due to the dynamic behaviour of vocabularies, a custom caching solution using hashes over datasets that invalidate cached values was implemented and proved to be useful.

In general, it showed that some optimisations with test data sets could increase performance by up to 55% in testing environments. These high results were not achieved in production with the whole data set though, also because the algorithm does many tasks, and only some of them can really be optimised. Implementing these optimisation strategies throughout the whole project did improve performance a lot though generating puzzles in less time.

The algorithms presented here are CPU-bound, not I/O-bound, so concurrency with multithreading was not considered. However, Python's multiprocessing seemed to be the right candidate for performance gain. Multiprocessing only allows improvement of data manipulation, thus not the whole algorithm can be parallelised. With multiprocessing being able to improve performance by a factor of n-1 cores, the performance gain is quite high for specific tasks when implemented correctly. To use multiprocessing, the whole data structure of the vocabularies had to be refactored to be able to copy the data to multiple cores. No recursive data is allowed, and implementing the singleton pattern proved to be necessary. The different and unforeseeable behaviours observed in multiprocessing between operating systems and the complexity of finding, proving and removing bugs lead to discarding all multiprocessing efforts. Other concurrency libraries like AsyncIO, joblib or Ray haven been considered but not been evaluated as the refactoring seemed too elaborate.

5. Evaluation

"True genius resides in the capacity for evaluation of uncertain, hazardous, and conflicting information."

- Winston Churchill

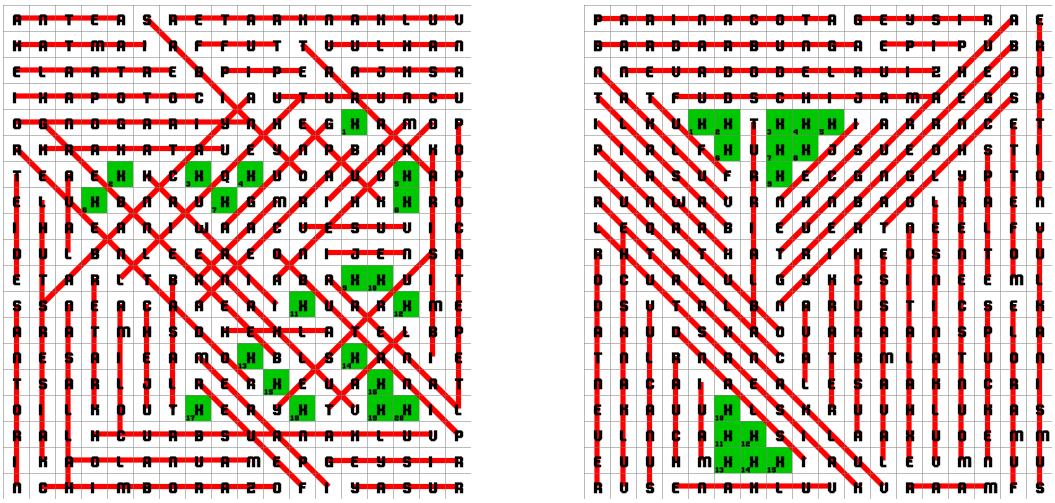
In this chapter, the results that were achieved in the work described in Chapter 4 are evaluated and discussed. While a conclusion which method to prefer is drawn in the next chapter, both approaches are evaluated individually first. Also, a discussion of the gained insights and a research roadmap for potential future work is presented.

5.1. Results

"Realists do not fear the results of their study."

- Fyodor Dostoevsky

This chapter evaluates the results of the scientific research trying to generate crossword puzzles with constraint programming and neural network. Since both approaches are unique in the methods used and the data structure, they are individually assessed first.



(a) Good Result of the Puzzle Algorithm

(b) Manually Rejected Puzzle

Figure 5.1.: Examples of Generated Puzzles. A good one on the left. The right one is manually rejected due to too many parallel lines and no nice spread between the solution word letters.

5.1.1. CSP

Some hardship cases do exist when using the algorithms in production. Analysing the vocabularies used in the generation process gives some useful insights. Initially, the average word length was expected to correlate with the complexity for the algorithm. However, it showed that the difficulty for the algorithm correlates more with a word length distribution. This means the more long words are available in a vocabulary, the harder it is for the algorithm to place enough words for a complete puzzle. On the other hand, it is easier to place the right amount of positions onto the grid with a vocabulary that contains more short words. Figure 5.2 shows three vocabularies in comparison, "Vulkane" which builds in an acceptable time, "OstseeBäder" which takes many more trials and "Bewerbung" which does not produce any usable results. It can be seen, that vocabularies with a skew towards longer words are increasing the runtime. Figures 5.3a and 5.3b show some more vocabularies in comparison to each other. Next to the vocabulary that did not produce any results, "Bewerbung", a vocabu-

lary with 86 words with an average word length of 11.76 letters, is an extension. Extending the vocabulary "Bewerbung" with 25 words with an average length of 5.8 letters created a new vocabulary with an average word length of 10.5 letters. With this extended vocabulary, it is possible to generate good puzzle results within seconds. This result contradicts with the initial thesis that vocabularies with an average word length of 8 or higher are hard to compile into puzzles.

The shown vocabulary v424 consists of all words extracted from the 424 manually created benchmark puzzles. While examining the character frequencies, as shown in Figure 5.4, no significant differences can be identified. There are two outliers, however. The first is the vocabulary "Tour". It consists of Japanese city names, containing the letter "A" very frequently and more than average letters "I" and "K" while containing very few letters "B" and "E". The second outlier is "Promi", which includes a list of celebrities names beginning with "B". These outliers do not correlate with the algorithms ability to compile a puzzle though since both vocabularies compile to high-quality puzzles within a short time. Comparing letter bigrams does not lead to any conclusions either, as no correlation to the puzzle results can be seen. Comparing bigrams of vocabularies is difficult as well because different vocabularies contain different bigrams and varying amounts of bigrams as seen when comparing "Staedtetour" (52 words, 117 bigrams) in Figure 5.5a with "Skat" (55 words, 173 bigrams) in Figure 5.5b. Only a correlation of the total amount of bigrams and overall size of the vocabulary can be assumed as seen in vocabulary "424" in Figure 5.5d.

Result

The following puzzle types can be generated with the introduced algorithm. A combination of the introduced GridGenerator and the GridFiller in the meta-algorithm "Word Search" can create word search puzzles with and without a solution word, as seen in figures 5.6a and 5.6b. Using the GridFiller stand-alone while providing it with a Swedish puzzle grid a Swedish puzzle can be generated as seen in

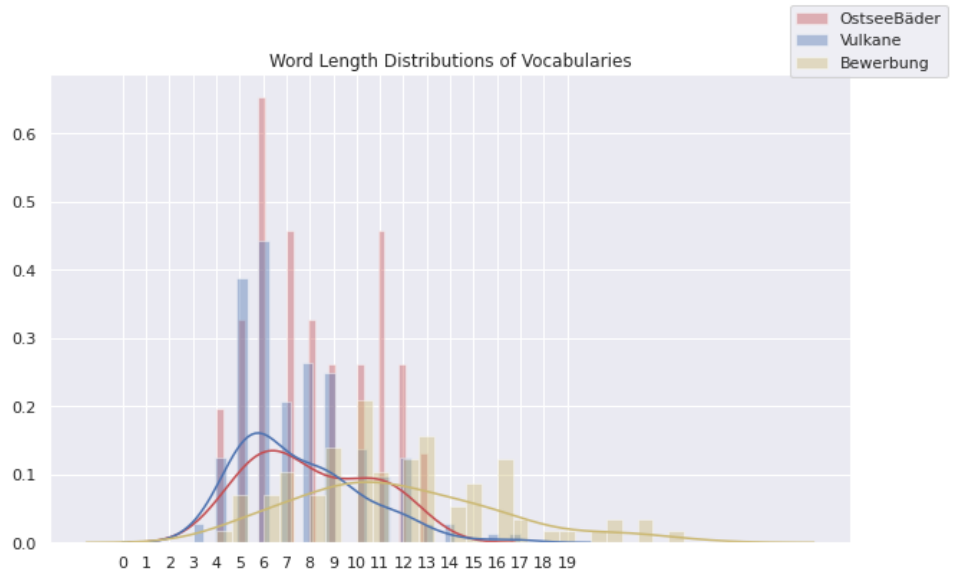


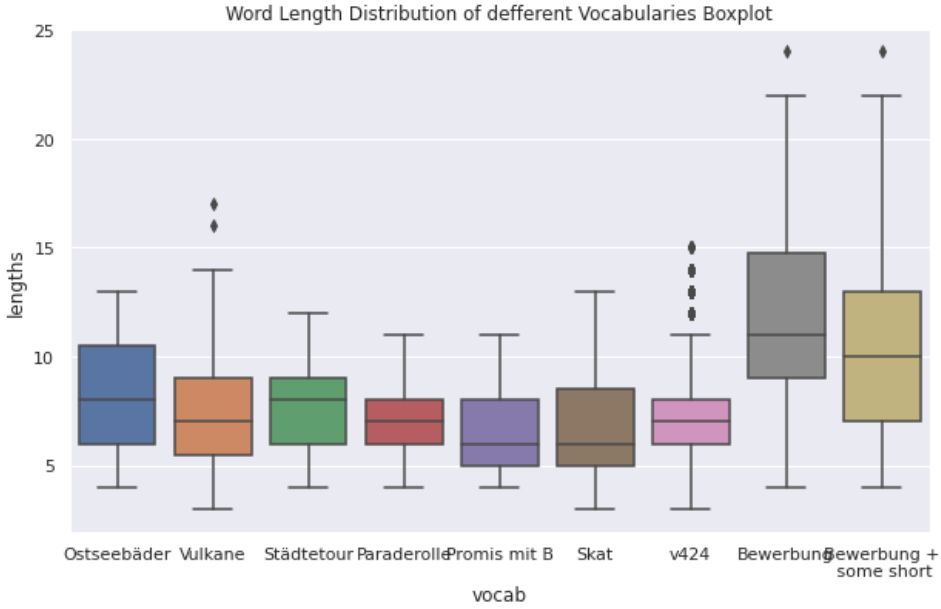
Figure 5.2.: Word length distribution to compare three vocabularies. The skew towards the left in the "Vulkane" vocabulary makes it easier for the algorithm to create usable results.

Figure 5.6c, although for a meta-algorithm "Swedish Puzzle" a specialized SwedishGridGenerator and an algorithm for clue assignment would need to be added, which was not part of this research.

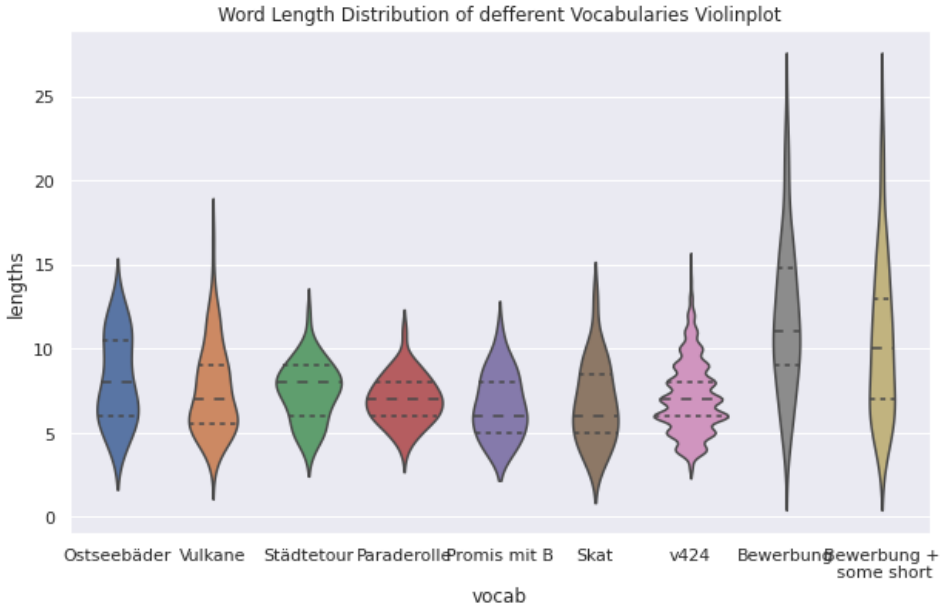
A combination of the GridExtractor and the GridFiller can be used to check for a single deterministic solution. This combination is not as crucial for the evaluated puzzle types but has shown to be usable as a quality measurement for puzzle types requiring a deterministic solution.

The introduced algorithms GridGenerator and GridFiller can respect given words or letters (Requirement 3.2.4) as well as restricted fields (Requirement 3.2.1). The GridGenerator can also respect free fields and thus create a shaped puzzle (Requirement 3.2.1). An example of a shaped puzzle with a given word can be seen in Figure 5.7.

It has shown that the vocab lengths distribution introduced in section 4.1.1 is best suitable. The mode "random" is creating better and



(a) Word Length Distribution Box Plot



(b) Word Length Distribution Violin Plot

Figure 5.3.: Word length distribution of multiple different vocabularies to evaluate the algorithm settings. The more short words a vocabulary contains, the better the algorithm performed, it created better results quicker.

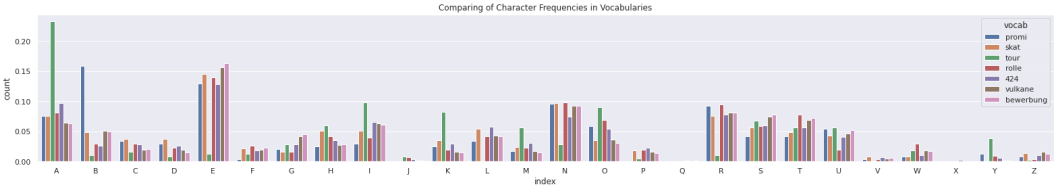
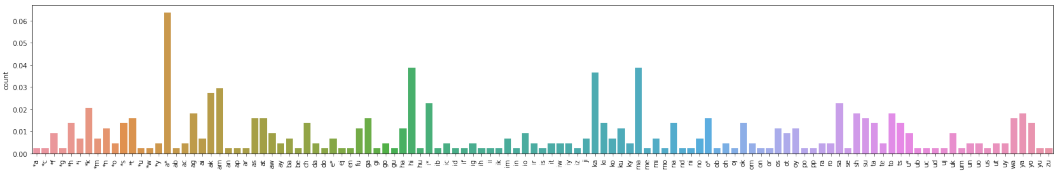
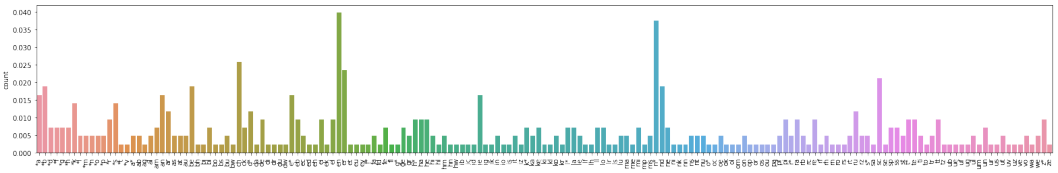


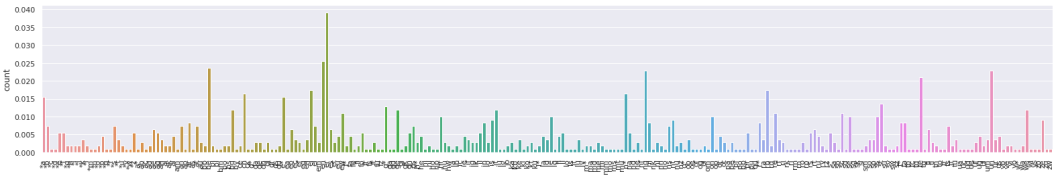
Figure 5.4.: Comparing character frequencies does not reveal any helpful insights either. The outliers did not show any different performance than the others.



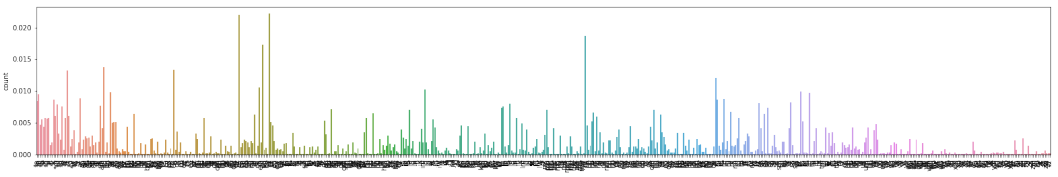
(a) Bigrams Staedtetour



(b) Bigrams Skat

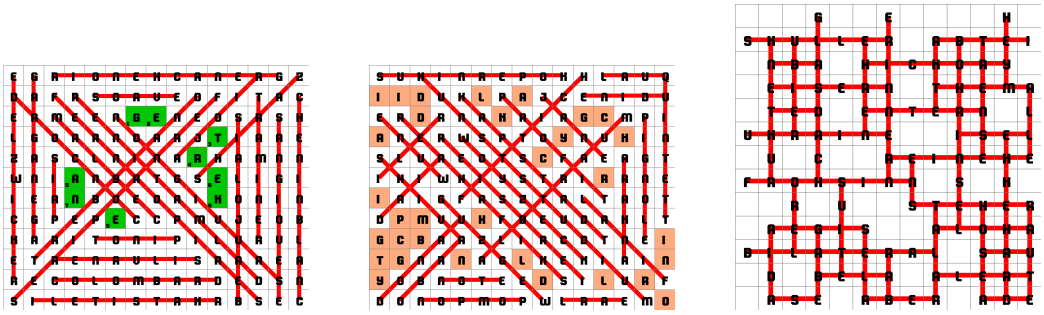


(c) Bigrams Bewerbung



(d) Bigrams 424

Figure 5.5.: Comparing the distribution of bigrams in different vocabularies does not reveal any helpful insights. No relevant connection between bigram distribution and algorithm performance could be detected.



(a) Word Search with Solution Word (b) Letter Desert without Solution Word (c) Swedish Puzzle

Figure 5.6.: Puzzles created by the introduced algorithms, a word search puzzle, a letter desert puzzle, and a Swedish puzzle.

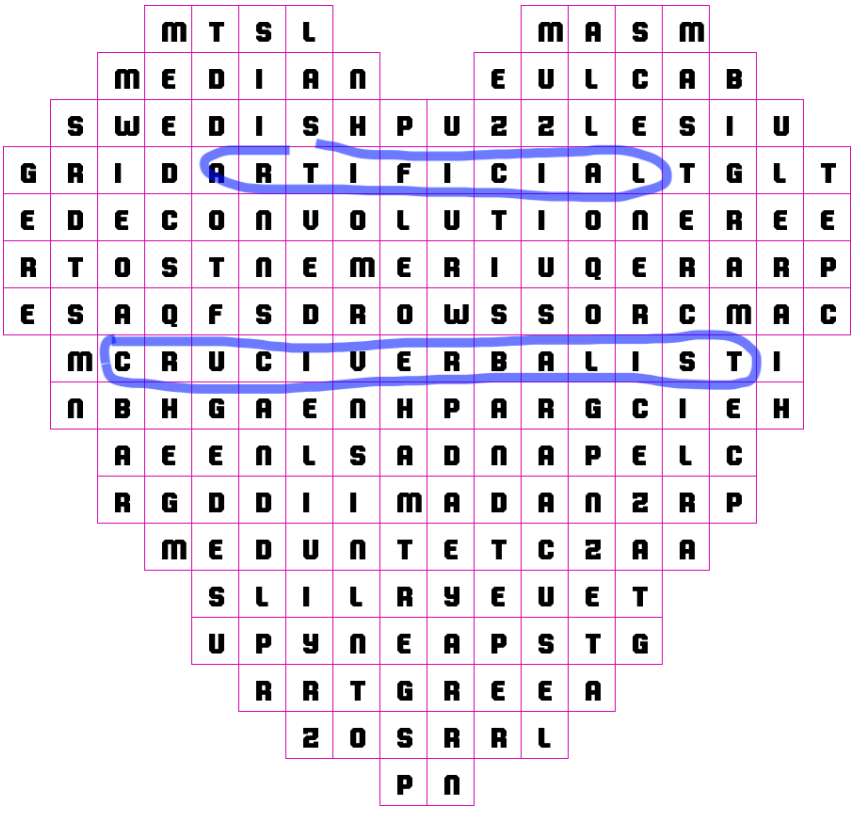


Figure 5.7.: A puzzle created by the introduced algorithms with a given shape and preassigned words (circled in blue).

faster results with large vocabularies, while the mode "decreasing" is necessary for smaller vocabularies to generate results at all. Phase three of the GridGenerator is currently not very helpful, as it creates too many overlapping issues, but has also shown not to be necessary and is thus not in use. It could be improved in future though, to respect overlap counters of positions and field population.

To give the GridFiller more opportunities for smaller vocabularies, each word in the vocabulary is initially reversed and added to the vocabulary while maintaining a blacklist forbidding reversed words. The algorithm then ignores the predefined direction of the positions in generated grids and tries to create a puzzle by placing words in both directions. This approach was chosen as it was easiest to implement and allowed to quickly prove the theory that giving the GridFiller the choice of word direction would improve performance. Allowing the algorithm to try words in both directions not only decreased the runtime from start to good result, but it also enabled the algorithm to deliver results where strictly given word directions prevented it from generating puzzles at all.

Overall the algorithm can produce usable results and is already used in a production environment by Krupion GmbH.

5.1.2. Neural Networks

> *"There are neural networks that can build whole apps from scratch - so why are we teaching high school kids to code? "*
>
> - Vivienne Ming

All evaluations are based on calculating the model accuracy as described in Section 4.2.3. Different experiments have shown that very high validation accuracy is necessary to create usable results. (Validation accuracy is rounded to two decimal points in this text, but is

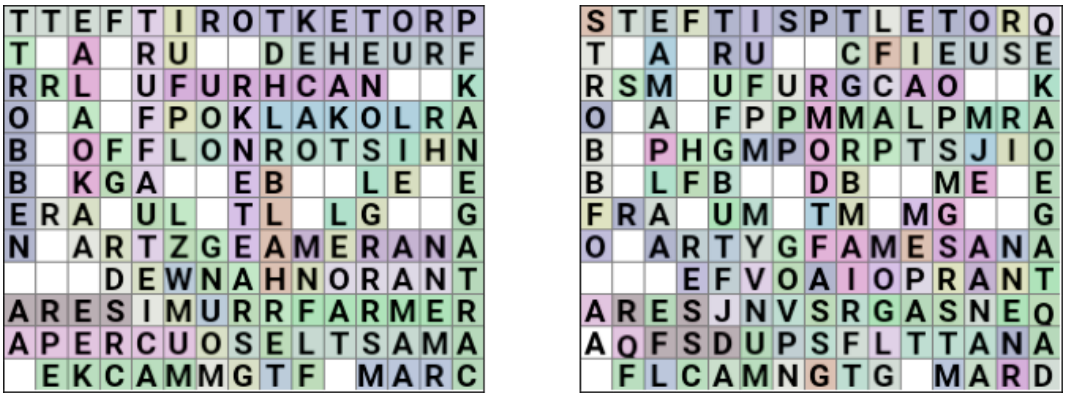
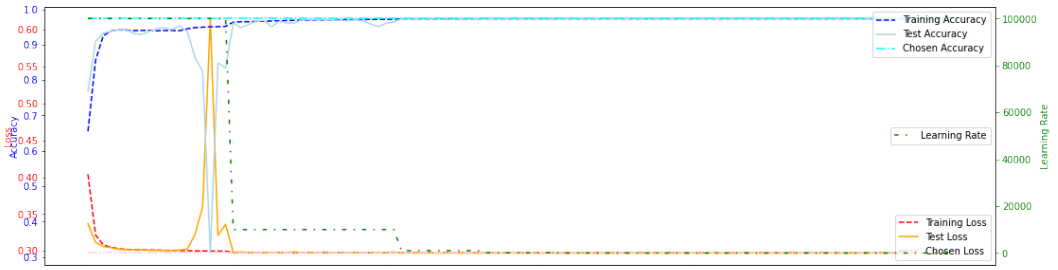


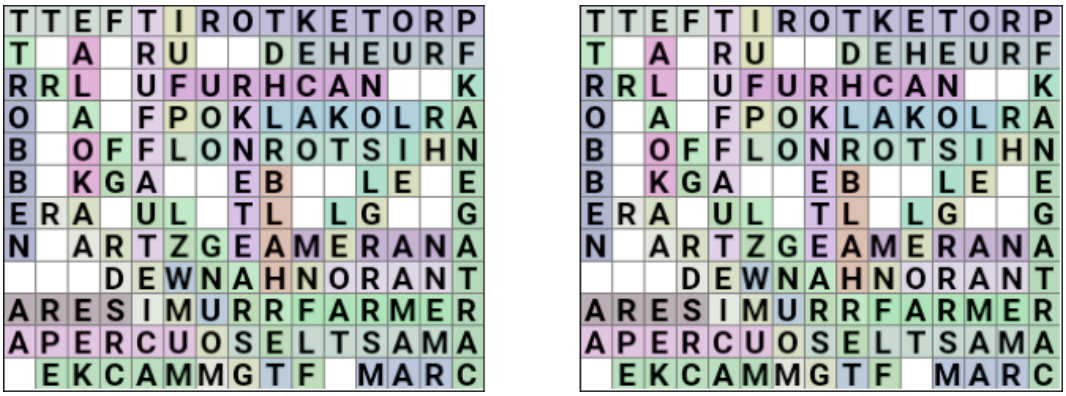
Figure 5.8.: The original puzzle (left) that is en- and decoded (right) with the introduced network architecture and a model trained to a validation accuracy of 97.84%. Already in the top row, four letters are wrong, not even considering the other two features. At least the empty fields are correctly assigned.

available with four decimal points as seen in Table 5.1 and the appendix.) As seen in Figure 5.8, a validation accuracy of 97.84% is not sufficient to auto encode a puzzle with good result. Too many fields are calculated wrong to recreate the original puzzle. Therefore, much higher accuracies are necessary to start with the puzzle generation through networks at all.

Contrary to expectations, deconvolutions on the decoder lead to weaker results than using convolutional layers. Also using concatenation instead of adding the output of previous layers to create the encoder did not improve performance. The architecture introduced in detail in section 4.2.3 is the best performing combination of hyperparameters for a convolutional puzzle autoencoder. It creates results with a validation accuracy between 98.85% and 99.88%. Figure 5.9a shows how training performed and that automatically adapting the learning rate does improve the results. Figure 5.9b showing the original puzzle on the left and the auto-encoded result on the right lets us conclude that the autoencoder architecture introduced is capable of en- and decoding puzzles. Table 5.1 shows the average validation accuracy of all tried convolutional architectures. Plots of the architectures, as well as the comparison of the other architec-



(a) Training history showing loss, accuracy, and learning rate over epochs trained.



(b) Original puzzle on the left and autoencoded puzzle on the right.

Figure 5.9.: The evaluation of the introduced deep convolutional puzzle autoencoder with training history and autoencoded results. It can be seen how the accuracy increases with more training epochs and with adapting the learning rate. The autoencoded puzzle has all letters correctly decoded.

tures, can be found in Chapter B. Sometimes, high validation results were achieved. However, it is necessary to account the number of epochs when the highest validation accuracy was achieved. A high validation after one epoch that continually dropped after proceeding training epochs should receive less attention. An example of dropping validation accuracy can be seen in Figure 5.10.

Seeing a decrease in accuracy when using pooling layers or when having lower dimensionality latent spaces in autoencoders (Table 5.1) can be interpreted in the following: The puzzle fields have a much

Architecture	Speciality	Val. Acc.	Epochs
CAE 0 (B.6)	Initial intuition	0.9577	63
CAE 0.5 (B.7)	MaxPooling	0.7998	40
CAE 1 (B.8)	Less depth	0.9473	10
CAE 2 (B.9)	Conv. instead of Dense	0.9684	4.5
CAE 3 (B.10)	Dropout	0.9410	17
CAE 4 (B.11)	Bigger Filters	0.9507	53
CAE 5 (B.12)	Smaller Filters	0.8450	6
CAE 6 (B.13)	No Batchnorm	0.9390	70
CAE 7 (B.14)	- and | Filters	0.9031	19
CAE 8 (B.15)	- | Filters Added	0.9943	34
CAE 8.5 (B.16)	- | Filters w. DeConv	0.9229	1
CAE 8.6 (B.17)	- | Filters no Adding	0.8297	12
CAE 9 (B.18)	- | Filters w. Concat	0.8177	3

Table 5.1.: Evaluation of the compared convolutional neural network architectures. The validation accuracy is an average over multiple trainings. Epochs is an average of the number of epochs after which the highest accuracy was reached.

higher information content compared to the pixels in images that have higher entropy. Pixels in images can have one of 16,777,216 different values where values close to each other do not necessarily make a visible difference allowing for some variation and inaccuracy. Pixels next to each other also often have similar values which means that a single pixel that is not 100% accurate does not render the whole image as wrong. For puzzles, this is different as each field can only have one of 26 values and neighbouring fields usually do not have the same value. A small change in the value of the field can leave a whole word, and thus the whole puzzle as invalid.

In the conducted experiments, it was not possible to find an architec-

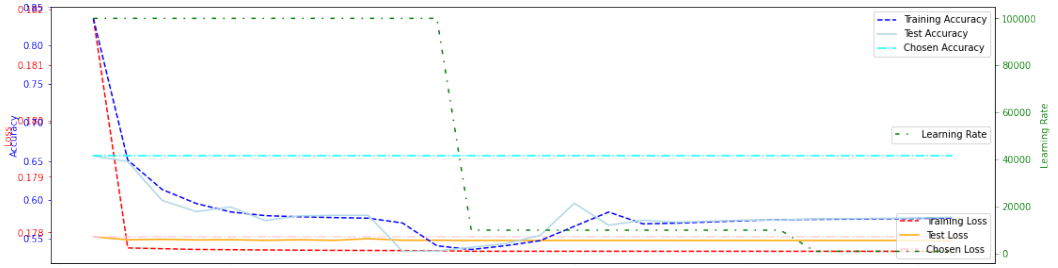


Figure 5.10.: An example for neural network training where the highest validation accuracy was achieved after the first training epoch. After epoch one the validation accuracy continually dropped. This result should therefore be given less attention.

ture that can autoencode vocabularies with high enough accuracy. Thus the overall goal of creating a neural network capable of learning puzzle structures to generate puzzles from given vocabularies was not achieved.

5.2. Discussion

> *"The aim of argument, or of discussion, should not be victory, but progress."*
>
> - Joseph Joubert

While the CSP approach creates valid and usable results, the neural network architectures evaluated here do not. Many of the requirements can easily be implemented as constraints but cannot be created for neural networks. While the neural network remains a black box, maybe learning the constraints, it cannot be proven that the requirements are correctly learned. The CSP can consider blacklists as constraints; the neural networks would have no chance to do so. While the blacklists could be implemented as QA checks discarding puzzles with mutually excluding words, this would lead to high reject

rates. Also, the necessity to have an extensive training set for each puzzle type and dimension makes neural networks rather impractical compared to the CSP approach that does not need any initial training data, but only the rules to be modelled. Another observation is that the results of the CSP algorithm are precisely calculated, if a solution exists, while the results of a neural network remain a statistical approximation [9]. This approximated result needs to be handled with care as faulty results can have serious implications depending on the domain of the problem. Noteworthy is that the CSP algorithm created the training set for the neural networks in this research. Hence, it is expected that the results of the former algorithm can be at most as good as those of the latter. Thus, the comparison does not make sense. Especially the question "Which approach generates better puzzles" is not answered as the results could be misleading and possible conclusions could be wrong. Hence, both approaches were evaluated individually concerning the quality of their results but were not compared to each other. In general, it can be concluded, that modelling puzzle generation as a constraint satisfaction problem delivers good results while training neural networks is rather impractical.

5.3. Research Roadmap

> *"A good plan violently executed now is better than a perfect plan executed next week."*
>
> - George S. Patton

The machine learning method evaluated in this thesis is not able to produce acceptable results. There are many more fields of research that could not be taken into account in this work. While not everything could be examined, there are other methods worth looking at. For example, deeper and wider networks should be evaluated. If a way can be found that allows for successful decoding of vocabularies other

neural network architectures like generative adversarial networks, especially conditional GANs, variational autoencoders or neural style transfer should be taken into account. Also, text-to-image systems like StyleGAN [95] or StackGAN [189] should be taken into account. Another machine learning method that could be evaluated is reinforcement learning where rewards could be given for correctly placing words. Other, completely different approaches like attentive neural processes or Neural Turing machines, especially differentiable neural computers, could also be tested, where the given vocabulary is stored in the memory used by the neural network. With the current state-of-the-art, it is expected that all machine learning approaches will still have the problem of being a black box making some requirements, like word blacklists, hard to implement. Although modelling blacklists for conditional generative adversarial networks [119] should be evaluated as well.

Another approach could be to simplify the problem. Already seen in the CSP approach, splitting the generation of grids and filling them reduced the complexity of each step dramatically. A similar approach could be tried with neural networks as well. For example, a generative network could be used to generate line patterns that are usable as grids. Another possible approach would be reducing the complexity of the data: instead of trying to place words to create a puzzle, the focus could be set on only placing letters. This could reduce the number of features from one to three. However, it must be remembered that this approach will probably decrease the entropy of the puzzles and thus increase the significance of the puzzle field values.

Since the puzzle fields have shown to have high information content, very high accuracy is necessary to decode puzzles. Neural networks return the probability of a letter, so there is always the chance of a field being wrong. A single wrong letter would leave a puzzle invalid. If many letters are correct, but only a few are wrong, a correction algorithm could be implemented as an information-theoretical error correction scheme. It could scan through the letters and compare letter combinations to the words in the vocabulary to repair broken words by fixing single out-of-the-line letters. Which word to use

could be decided with the Levenstein's distance [186] between letter combinations and vocabulary words.

Also, other existing algorithmic areas like genetic or memetic algorithms [114, 120] could be focused on. While a custom backtracking algorithm was used in the CSP approach for generating grids, this could be adapted to place words immediately instead of lines only. Maybe maze generation algorithms could be used, or grids could be modelled as graphs, where letters are modelled as nodes and paths throughout the graphs model words. Then a minimal path algorithm or similar could find a graph that can be placed on the given grid.

In case of a custom word placing algorithm, utilizing a recommender system could be evaluated. A properly trained or calculated recommender system could suggest words that can be placed next. The recommender system could take semantic features as well as letter positions as features or a combination of both into account. A similar approach could be to train a word embedding that takes letters into account, placing words that can intersect with each other closer in the space. However, any approach where words have to be learned could make it difficult to use new words that were not present during training.

Extending the constraint satisfaction problem (CSP) with an associated objective function and solving it as a constraint optimisation problem (COP) could lead to better results. The objective could take the overlap factor of words, the deviation of the solution word length, the use of blacklisted words and the distribution of word orientations into account. While the current CSP creates a result, if one exists, the COP could return the best solution of all existing solutions. If the search for the optimal solution meets the timing requirements (Requirement 3.2.4) needs to be evaluated.

6. Conclusions

> *“Three things cannot be long hidden: the sun, the moon, and the truth.“‘*
>
> - Buddha

The results achieved with the CSP approach is rated as successful by the domain experts Stefan Krüger and his team at Krupion GmbH. A puzzle generator service has been implemented and integrated into the existing puzzle frontend software, as described in Chapter 3.3 "Expectations".

The objective of creating a puzzle algorithm, the "Artificial Cruviverbalist", was achieved in the form of a modular system that offers multiple algorithms via a REST API. Contrary to the initial assumption, *neural networks* currently do not seem to be feasible for puzzle generation. For neural networks an extensive set of training data is needed and the difficulty of implementing some requirements, like blacklists or flexible puzzle sizes, make neural networks unsuitable. Most requirements, for example crossing words and valid assignments for filling grids, can be defined as constraints for a *constraint satisfaction problem*. Some other requirements have shown to be more suitable for implementation in a *custom backtracking algorithm* to create puzzle grids. Splitting the generation of grids and filling them in a separate step has shown to produce excellent results. Combining backtracking in custom algorithms and in constraint satisfaction programming ensures that a solution is found, if one exists. Another vital insight gained is the importance of word length distribution in vocabularies. The more short words a vocabulary contains, the better are the results of the puzzle generating process. Contrary to initial beliefs,

letter and bigram frequencies of the used words do not seem to have a significant impact on the generation process.

When comparing the introduced methods for puzzle creation, the following can be concluded: Training requirements implicitly with *machine learning* (black box) is not as suitable as specifying them explicitly as constraints for *constraint programming* with current state-of-the-art tools. Also, the high information content of the separate chunks of information and the necessity of exact results makes neural networks too inaccurate. The general rule that can be concluded: it is critically important to select the right model for the approached task. Depending on the domain and the necessary accuracy, choosing the right method can have a high impact on the results and applicability of the implemented system. A puzzle neural network only approximates the probability of a letter placed in each field. Even if the result has an accuracy of 99.8%, an error with the probability of 0.2% can occur invalidating the whole result. This need not be a problem for many applications, but can lead to severe implications that are hard or impossible to detect. While quality checks and automatic error correction in the puzzle creation process can discard or repair invalid results, other problems do not forgive such errors so easily. An example of adverse outcomes of very unlikely inaccuracies with dramatic consequences is the bug in Xerox scanners that alters numbers while photocopying documents, as revealed by David Kriesel [49]. A change of a single number in a copied document could have severe implications like wrong invoices, construction plans with inaccurate measurements or even incorrect dosage of medicine. Also, the impact of possibly faulty results needs to be taken into account when selecting a model and method. While a broken puzzle might ruin someone's leisure time, the wrong medication could lead to fatalities, and thus exact results are much more critical. When exact results are needed, explicit calculations with constraint programming are highly recommended over statistical approximation with neural networks when possible.

6.1. Future Work

> *"Success is not final, failure is not fatal: it is the courage to continue that counts."*
>
> - Winston Churchill

Further scientific research which can be done, especially with neural networks, is described in detail in the research roadmap in Section 5.3. Directly following this work, some improvements to the *constraint satisfaction programming* (CSP) algorithm will be made. Additional constraints will be implemented, for example, improving the blacklist constraint generation that breaks the timing requirement (Requirement 3.2.4) for large blacklists in its current implementation. It will be evaluated if extending the CSP with an optimization function and solving it as a *constraint optimisation problem* (COP) can create better results. Adaptations to the introduced algorithms to create further puzzle types will be made: One of the first steps aims at switching letters for syllables or bigrams to enable the GridFiller algorithm to create syllable puzzles. However, also an adaptation of the GridGenerator to create Swedish puzzles is planned. To fully generate Swedish puzzles, additional sub-algorithms are needed to create and fill in question fields.

The current implementation offers a REST API that keeps the connection between client and server open. It makes sense to introduce a job queue so that the HTTP connection can be closed earlier during long-running compilation processes. The initial request will then contain a callback address, which is called after the compilation process, to return the created puzzle.

An eye will be kept on the developments in deep learning, but in the near future, constraint satisfaction programming will be the choice for generating puzzles.

Appendix

Appendix A.

Additional Requirements

"The written word is the greatest sacred documentation."

- Lailah Gifty Akita, *Pearls of Wisdom: Great mind*

A.0.1. Requirements for Grids

- Swedish
  - Clue field: the field that contains the question
  - Blind field: Answer letter field that is only crossed / used by one word
  - Regular field: Answer letter field that is crossed / used by two words
  - If grid gives enough space, no words with one or two letters must be used
  - Minimum 50% of the letters of a word has to be crossed twice (Regular fields) (Examples: 3 letters: 2 crossings, 4 letters: 2 crossings, 5 letters: 3 crossings, 6 letters: 3 crossings, 7 letters: 4 crossings)
  - Divider (clue) fields (question fields with arrows) must not divide a grid in closed segments (graph) jedes feld von jedem erreichbar, muss durchgängiges system sein, graph knoten = buchstabenfeld

- Not more than four question-fields in a chain (mostly only one or two field groups, three or four field groups used rarely)
- No double blind fields (that are only crossed once) next to each other
- "good grid"-statistics:
  - 25% question fields
  - 12% blind fields
  - The rest are double crossed fields

  Blindfelder: buchstabenfeld das nur einmal gekreuzt zwei blindfelder nebeneinander verboten, wie ein undefiniertes
- Depending on grid and vocabulary these optimal values can differ
- Verknotungsfaktor
- mehr verknotung weniger fragenfelder -> schwerer
- Gitter abhängig vom Vokabular
- Gitter für leichten Wortschatz zugewiesen
- Gitter Wortlängen abhängig vom Vokabular

A.0.2. Requirements for Vocabularies

- Clue vocab
  - Currently manually created
  - One answer could have multiple different clues to choose from (e.g. answer "N", possible clues "North", "Stickstoff").
  - Use Wikipedia dumps (intro might be enough) to generate clues to words
- when the algorithm is asked to generate 500 puzzles the puzzles should not repeat words, so the available vocabulary needs to be updated dynamically during the creation process
- Language profiles
  - Each word is assigned to one or more topics.
  - A language profile can be seen as a group of words that has a linguistic and thematic weighting.

  - Language profiles are used to create thematic puzzles.
- Grids
  - Need to be uniquely identifiable
  - Should be comparable for example with a calculated edit distance
  - Prevent to be used multiple times in one run
  - In one run they should be fairly different
  - A grid can be rotated and transposed to create a new grid as seen in Section 4.3.7.
- Some puzzles types require a unique/deterministic solution
  - QA: Extract grid and vocabulary and feed both into the GridFiller, only one solution is allowed
- Puzzles need to be comparable by content (e.g. similarity based on used vocabulary)
- Use a constraint solver to check if puzzles are solvable

Appendix B.

Neural Net Architectures

This chapter gives an overview of the evaluated architectures and sets of hyperparameters for the neural networks. While tests have shown that a very high validation accuracy is necessary, focus was set on finding the best configuration of hyberparameters. All models were trained with the augmented data set described in section 4.2.2. After evaluating different optimizers and loss functions these architectures where trained with Adam optimizer, with an initial learning rate of 0.001, and binary cross-entropy loss function, as these have shown the best results. The tables show validation accuracy and the epoch with the highest accuracy selected by early stopping. Since random creation of batches which leads to varying results, both values are averages over multiple experiments.

B.1. Deep Neural Autoencoder for Puzzle Space

Deep neural autoencoders constructed of only dense layers. To feed the data into these, the data is flattened first. A comparison of the results can be seen in Table B.1.

Arch.	Speciality	Val. Acc.	Epochs
DAE 0 (B.1)	Initial intuition	0.839	139
DAE 1 (B.2)	Deeper	0.7431	207
DAE 2 (B.3)	Even Deeper	0.6617	1
DAE 3 (B.4)	Wider	0.8223	3
DAE 4 (B.5)	Dropout	0.8949	82

Table B.1.: Evaluation of the compared neural network architectures. The validation accuracy is an average over multiple trainings. The epoches is an average of the number of epoches after which the highest accuracy was reached

B.2. Convolutional Autoencoder for Puzzle Space

Deep convolutional autoencoders where the main focus was set on convolutional layers but to try different architectures also dense, dropout, batchnorm and deconvolutional layers have been considered. These autoencoders try to en- and decode the puzzle space. A comparison of the results can be seen in Table B.2.

B.3. Convolutional Autoencoder for Context Space

These autoencoders try to en- and decode the context space.

Arch.	Speciality	Val. Acc.	Epochs
CAE 0 (B.6)	Initial intuition	0.9577	63
CAE 0.5 (B.7)	MaxPooling	0.7998	40
CAE 1 (B.8)	Less depth	0.9473	10
CAE 2 (B.9)	Conv. instead of Dense	0.9684	4.5
CAE 3 (B.10)	Dropout	0.9410	17
CAE 4 (B.11)	Bigger Filters	0.9507	53
CAE 5 (B.12)	Smaller Filters	0.8450	6
CAE 6 (B.13)	No Batchnorm	0.9390	70
CAE 7 (B.14)	- and | Filters	0.9031	19
CAE 8 (B.15)	- | Filters Added	0.9943	34
CAE 8.5 (B.16)	- | Filters w. DeConv	0.9229	1
CAE 8.6 (B.17)	- | Filters no Adding	0.8297	12
CAE 9 (B.18)	- | Filters w. Concat	0.8177	3

Table B.2.: Evaluation of the compared convolutional neural network architectures. The validation accuracy is an average over multiple trainings. The epoches is an average of the number of epoches after which the highest accuracy was reached

B.4. Ideas to Generate Vocabularies

Even though the focus was not set on generating vocabularies, these ideas to create these arose.

- Analytic–synthetic distinction
- Elegant variation
- Synonym ring
- Antonyms
- Lexeme
- Morphemes / Monems

Arch.	Speciality	Val. Acc.	Epochs
CAE 10 (B.19)	Initial intuition	0.68	28
CAE 11 (B.20)	MaxPooling	0.65	1
CAE 12 (B.21)	Less depth	0.8035	1

Table B.3.: Evaluation of the compared convolutional neural network architectures for autoencoding the context space. The validation accuracy is an average over multiple trainings. The epoches is an average of the number of epoches after which the highest accuracy was reached

- Lemma
- Word embeddings
- Homophones
- Homographs
- Homonyms
- Rhymes
- Acronyms
- More -onyms can be found on Wikipedia[1]
- Maybe use WordNet[2] for english puzzles
- ConceptNet[3]
- Natural Language Toolkit[4]
- GraphWords[5]
- To build vocabularies maybe use Wikimedia Downloads[6] as source
- Automatically generate blacklists with word2vec, ELMo or BERT, word embeddings (e.g. family members: Mutti, Mama, Mutter, Vater, Opa)[7]

[1] https://en.wikipedia.org/wiki/-onym (Accessed on: 2020-05-24)
[2] https://wordnet.princeton.edu/ (Accessed on: 2020-05-24)
[3] http://conceptnet.io/ (Accessed on: 2020-05-24)
[4] http://www.nltk.org/ (Accessed on: 2020-05-24)
[5] https://graphwords.com (Accessed on: 2020-05-24)
[6] https://dumps.wikimedia.org/ (Accessed on: 2020-05-24)
[7] https://towardsdatascience.com/beyond-word-embeddings-part-2-word-vectors-nlp-modeling-from-bow-to-bert-4ebd4711d0ec (Accessed on:2019-09-27)

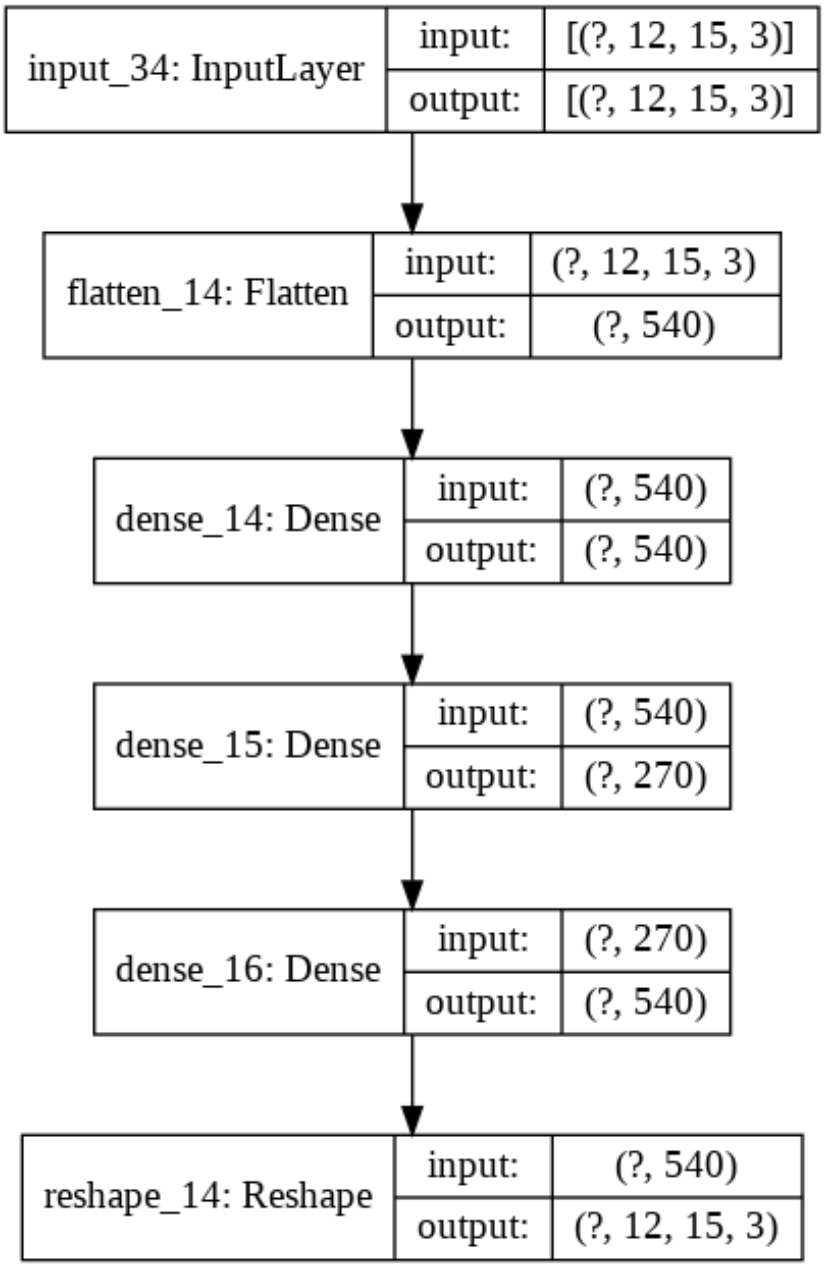


Figure B.1.: DAE 0

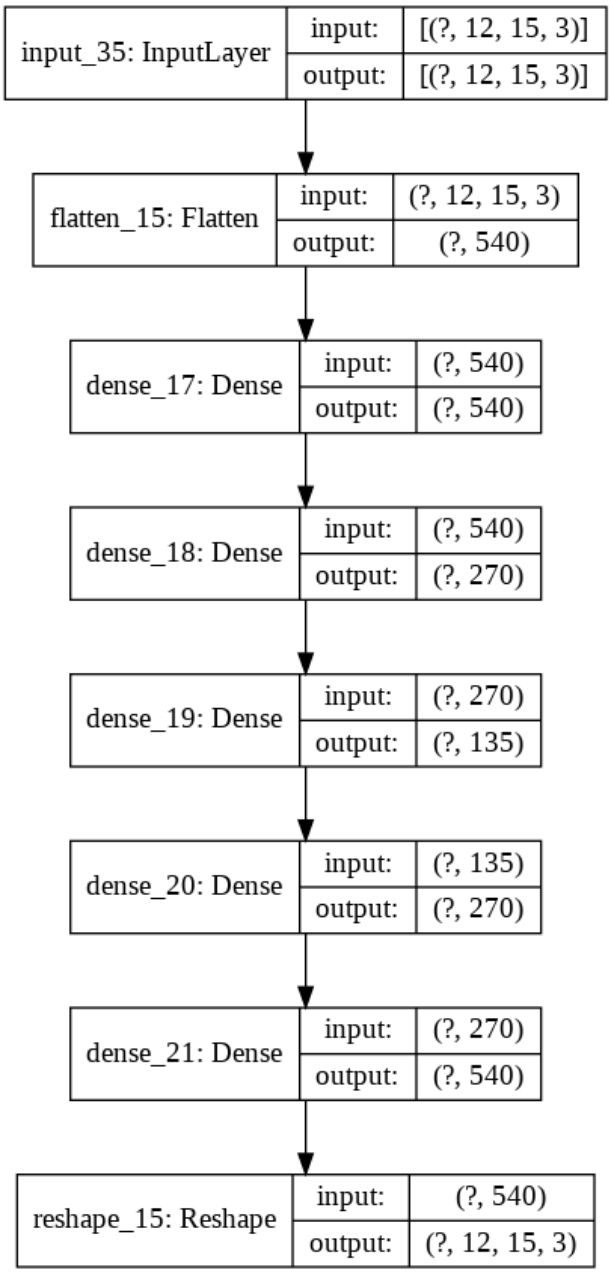


Figure B.2.: DAE 1

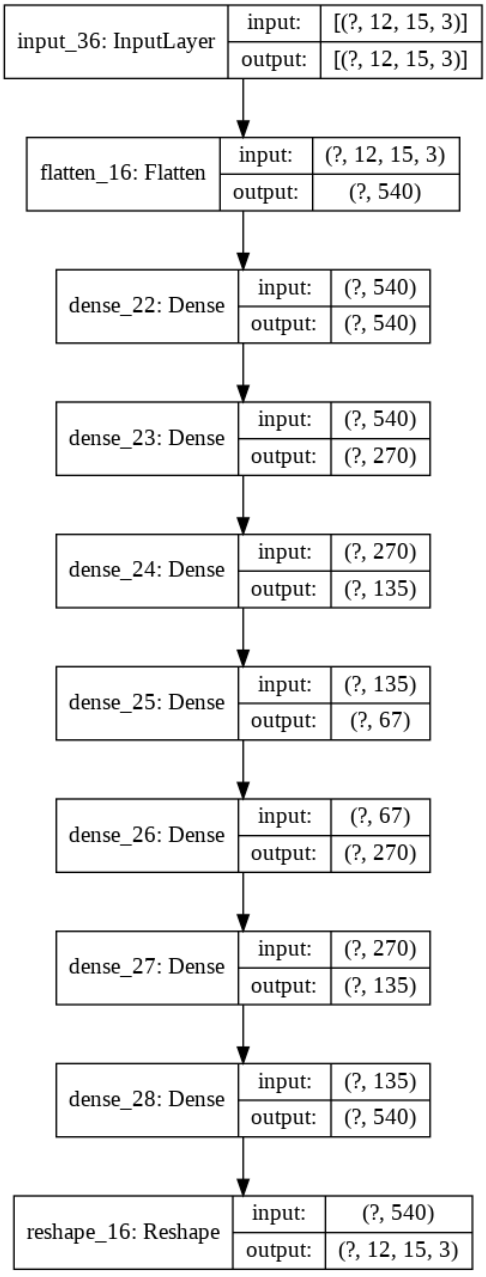


Figure B.3.: DAE 2

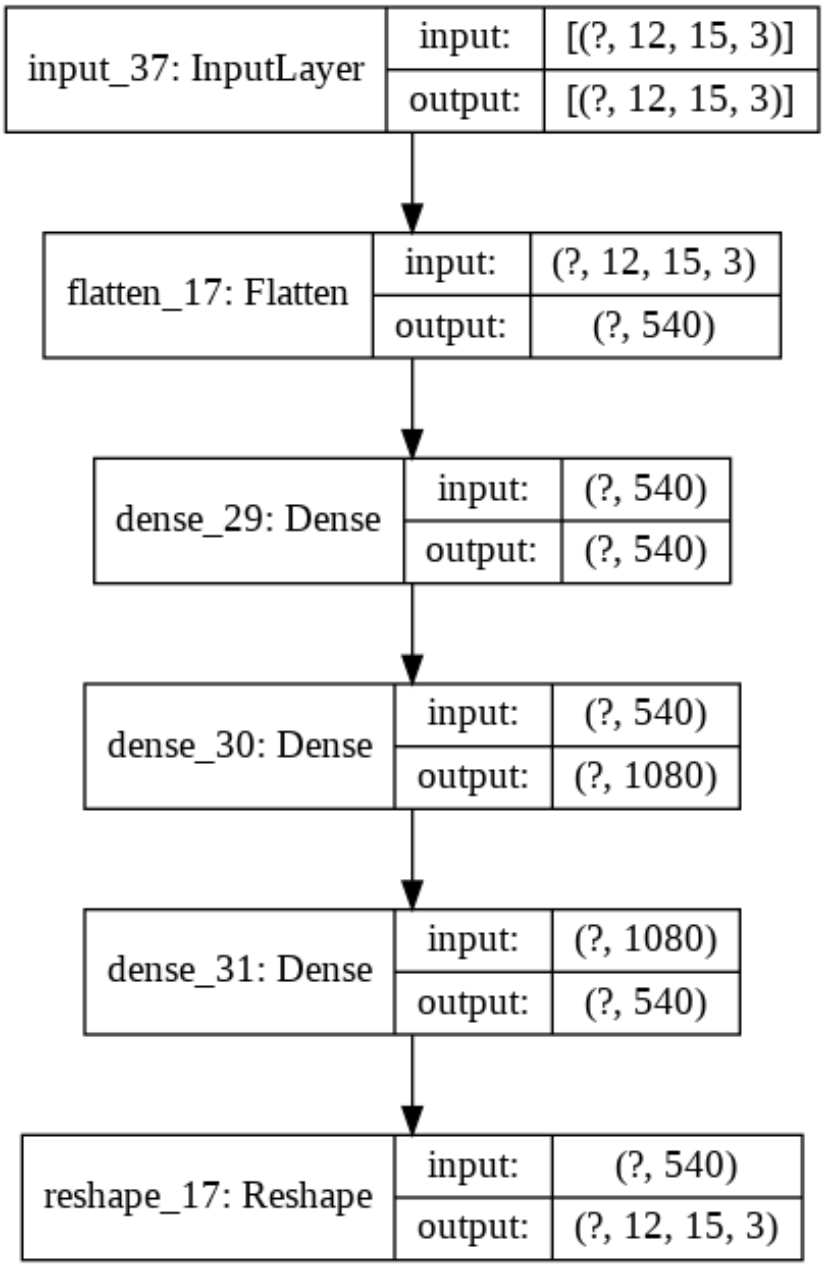


Figure B.4.: DAE 3

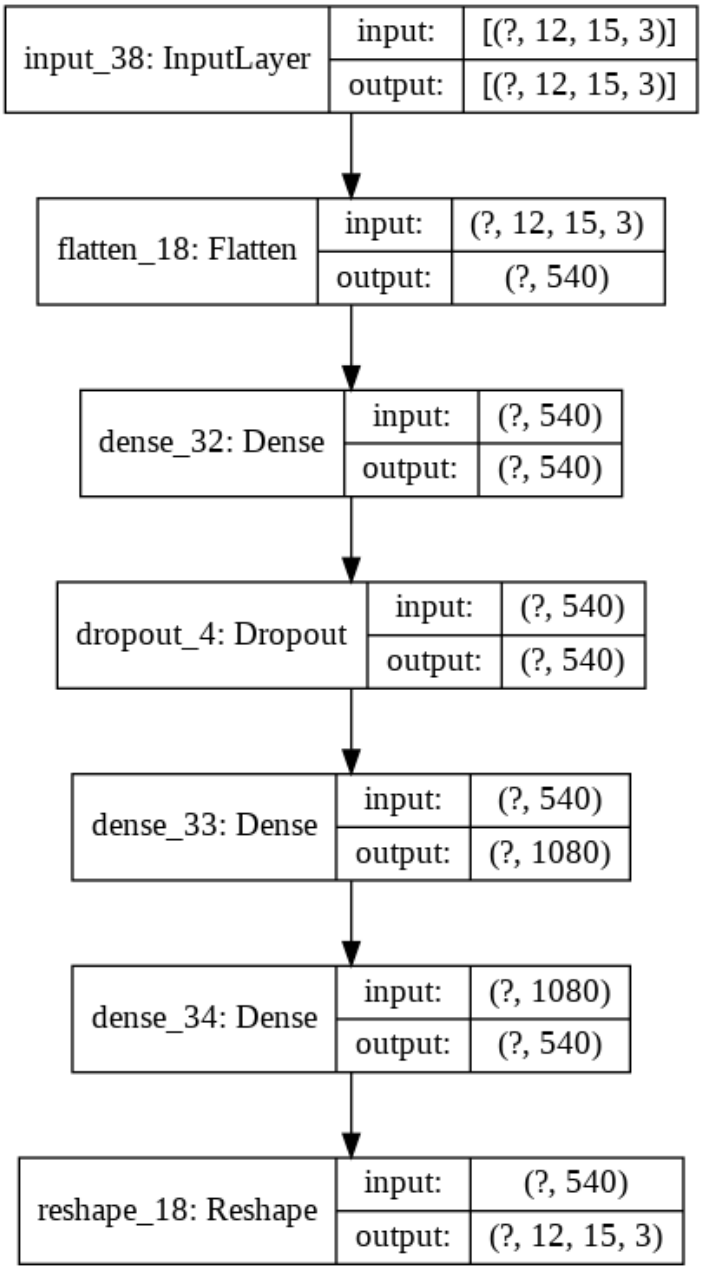


Figure B.5.: DAE 4

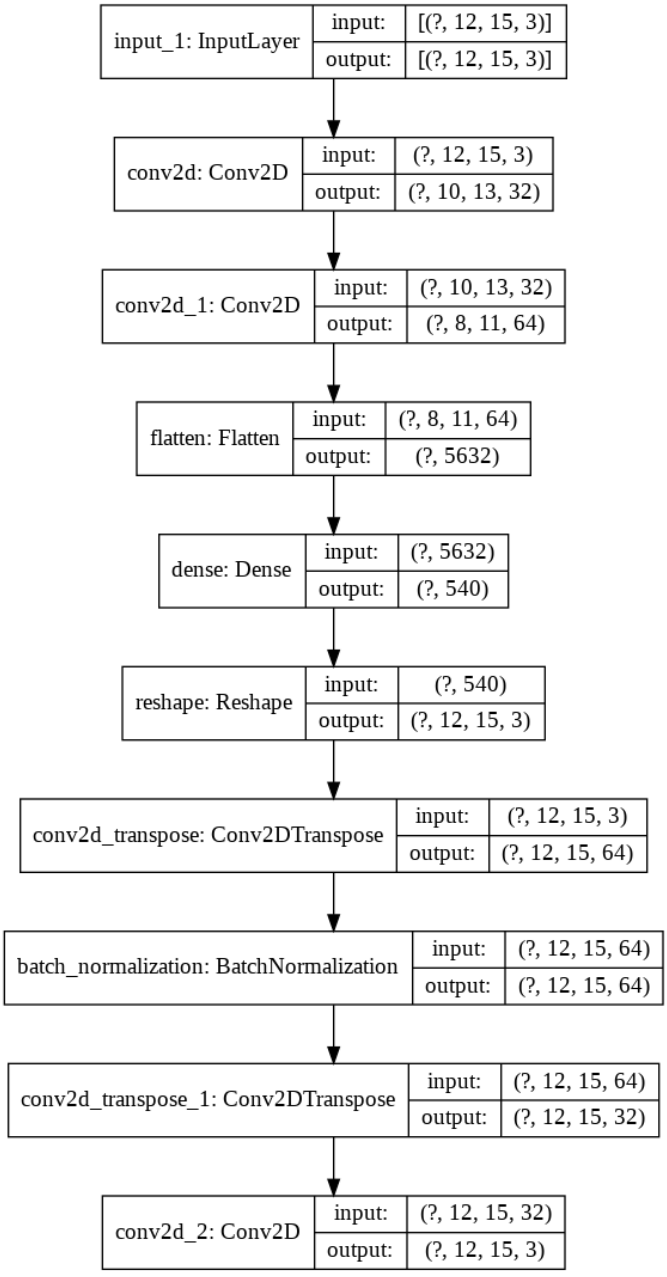


Figure B.6.: CAE 0

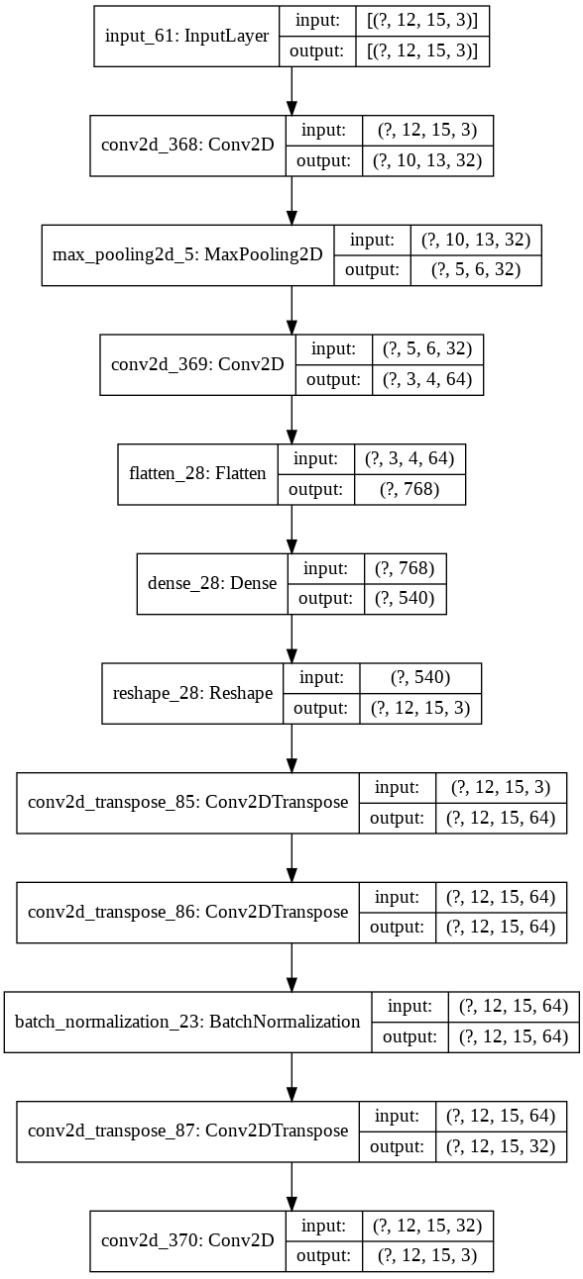


Figure B.7.: CAE 0.5

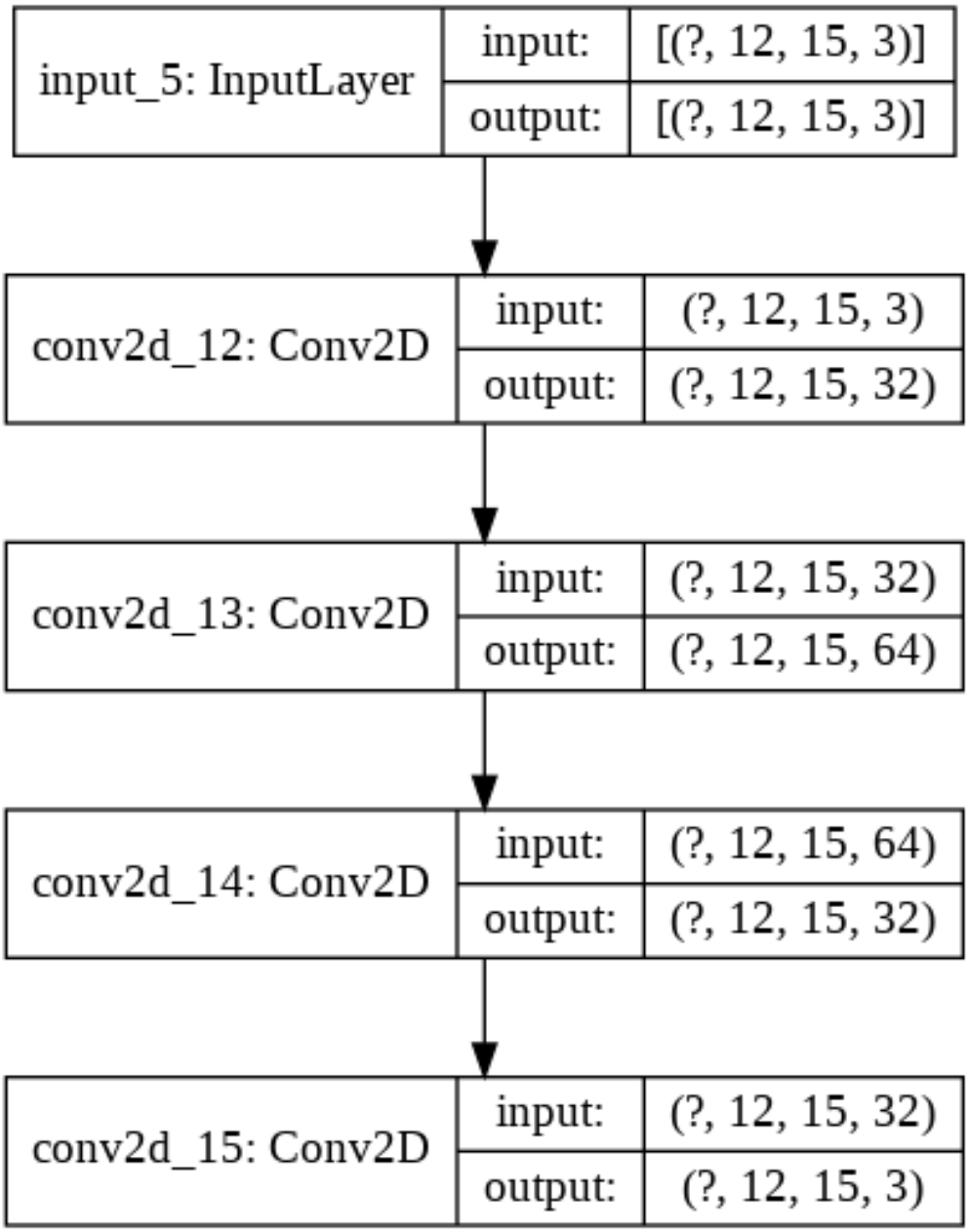


Figure B.8.: CAE 1

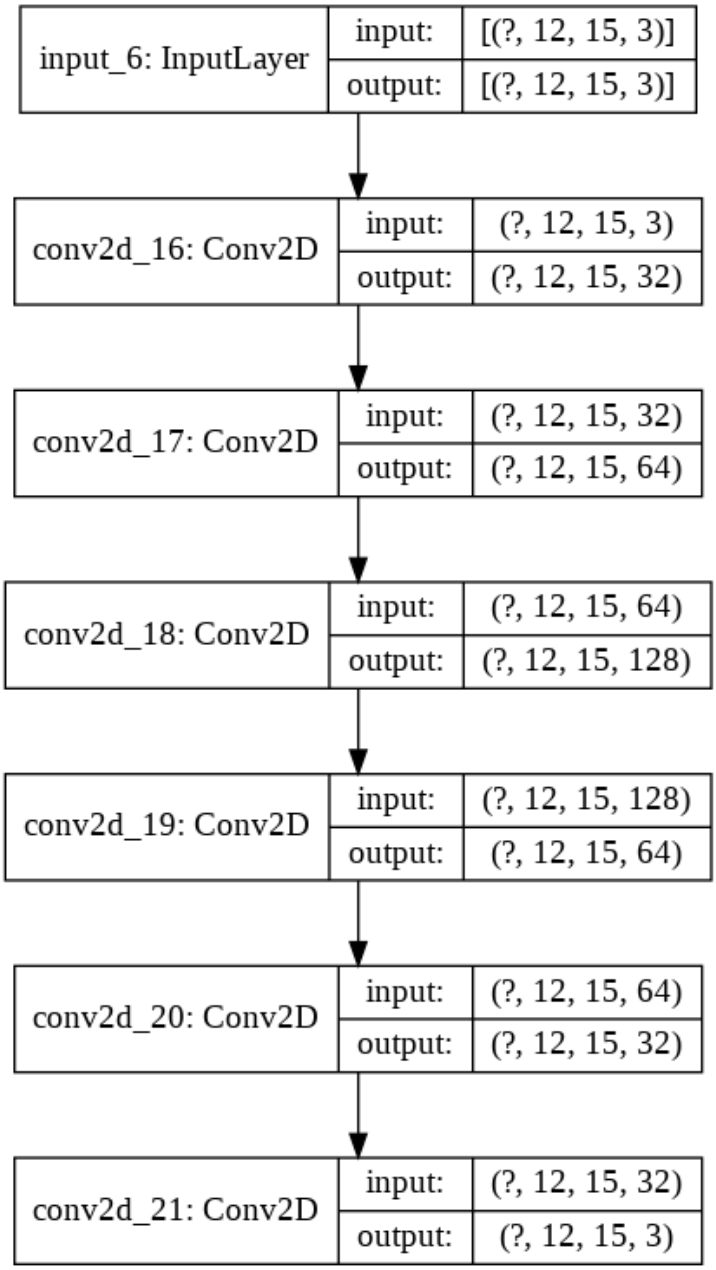


Figure B.9.: CAE 2

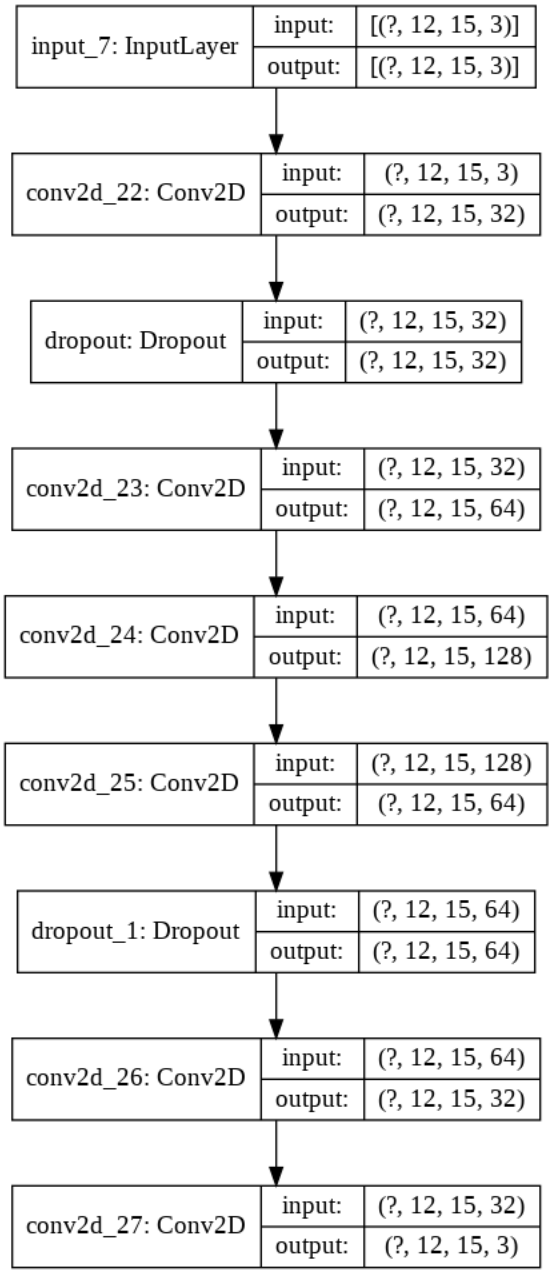


Figure B.10.: CAE 3

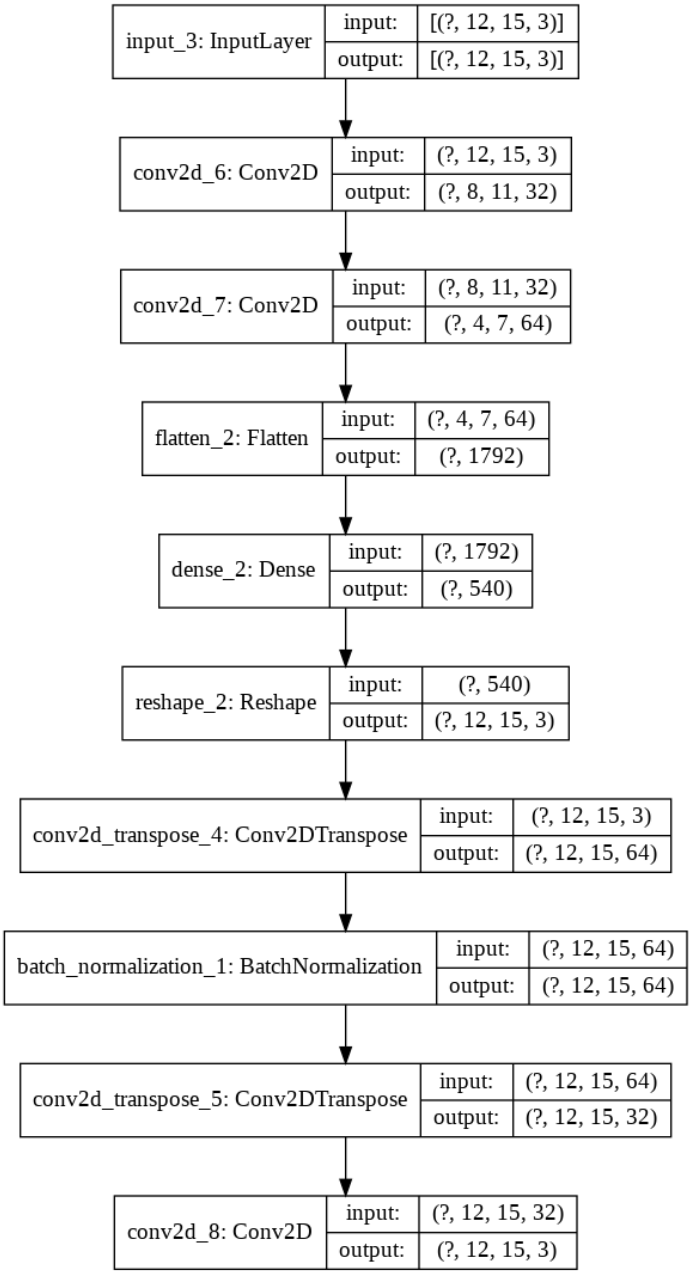


Figure B.11.: CAE 4

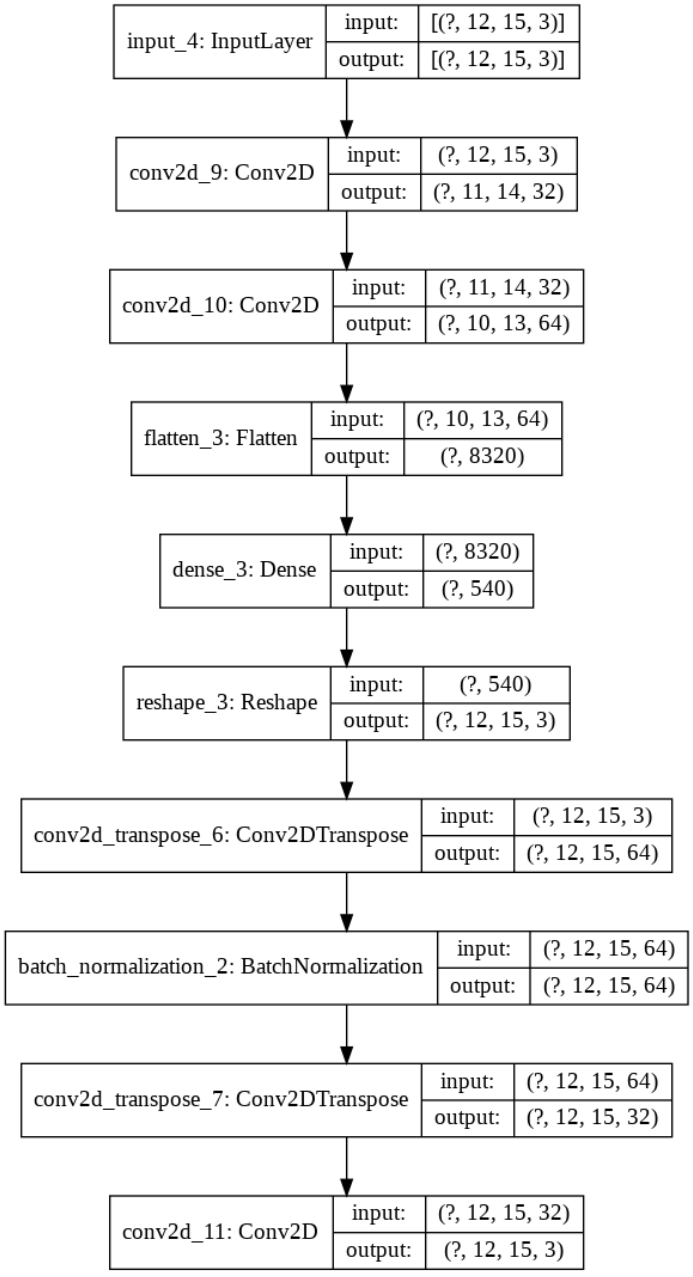


Figure B.12.: CAE 5

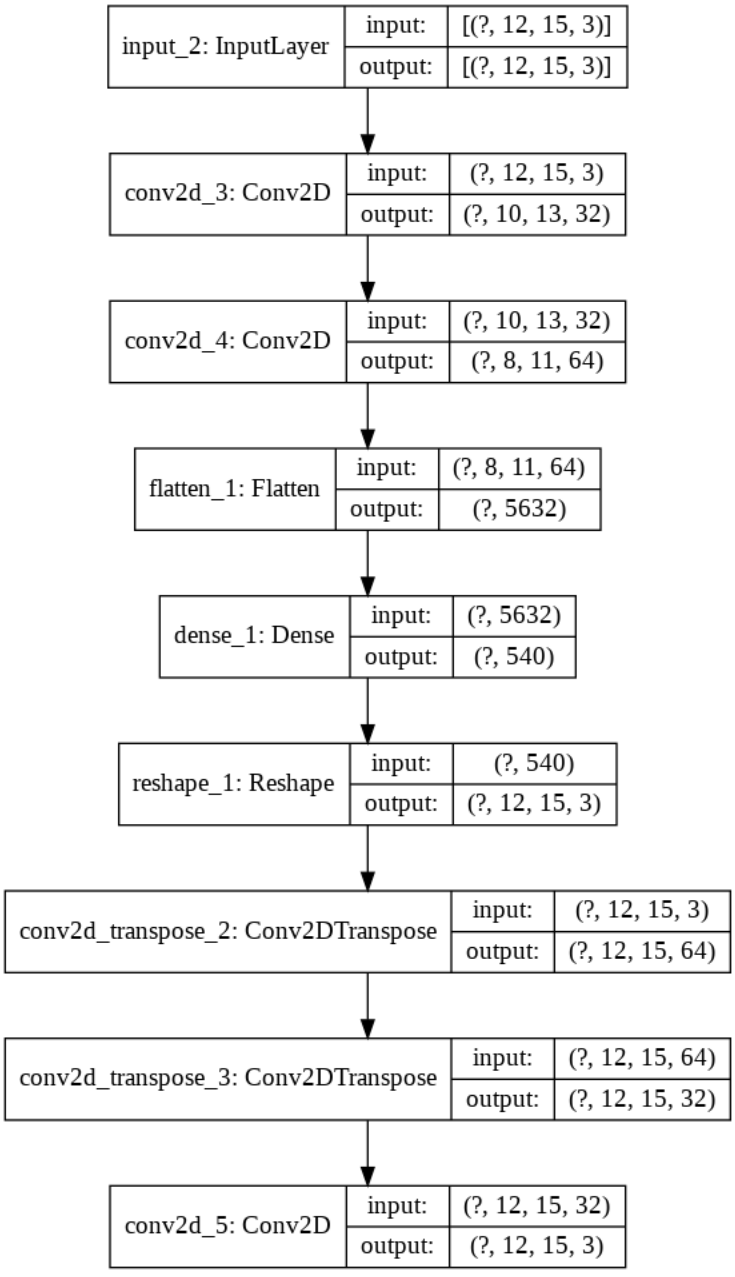


Figure B.13.: CAE 6

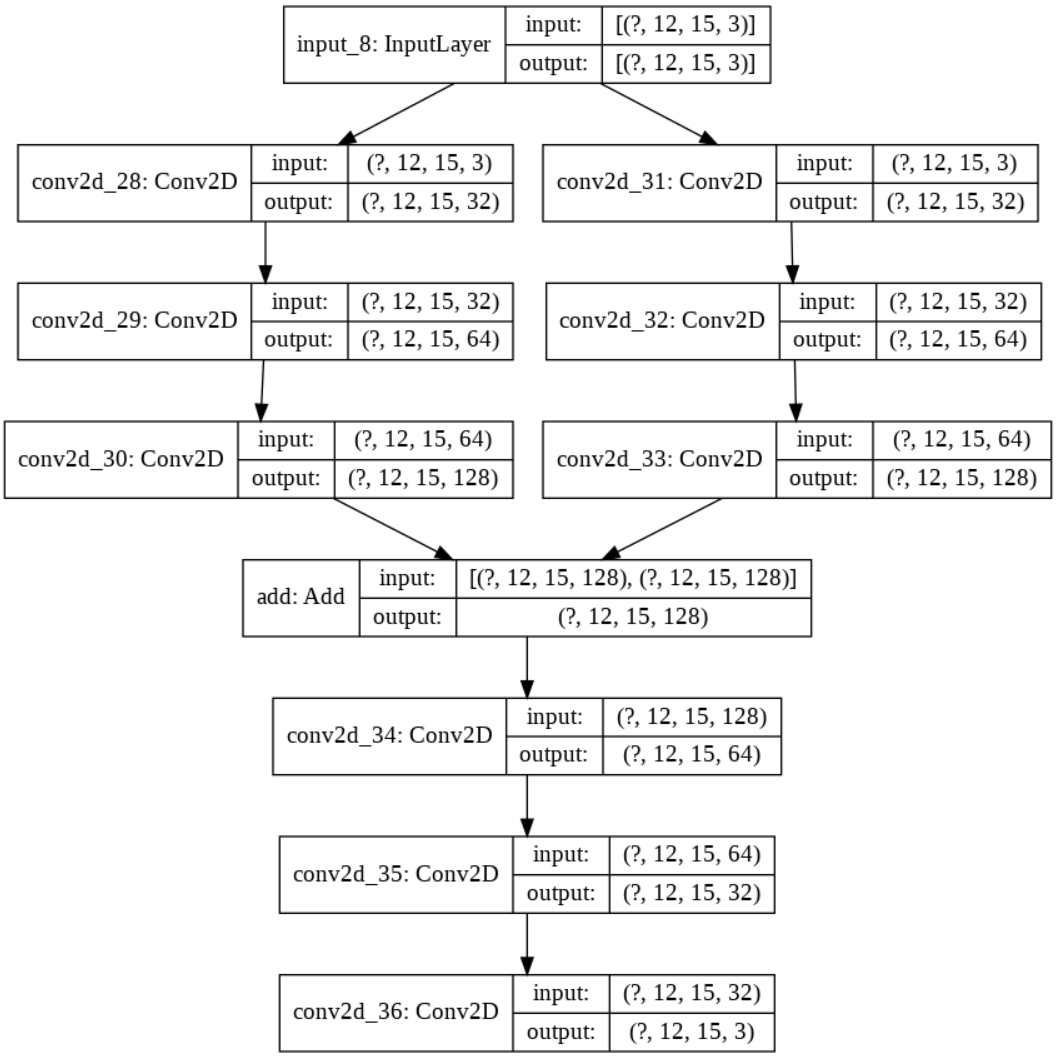


Figure B.14.: CAE 7

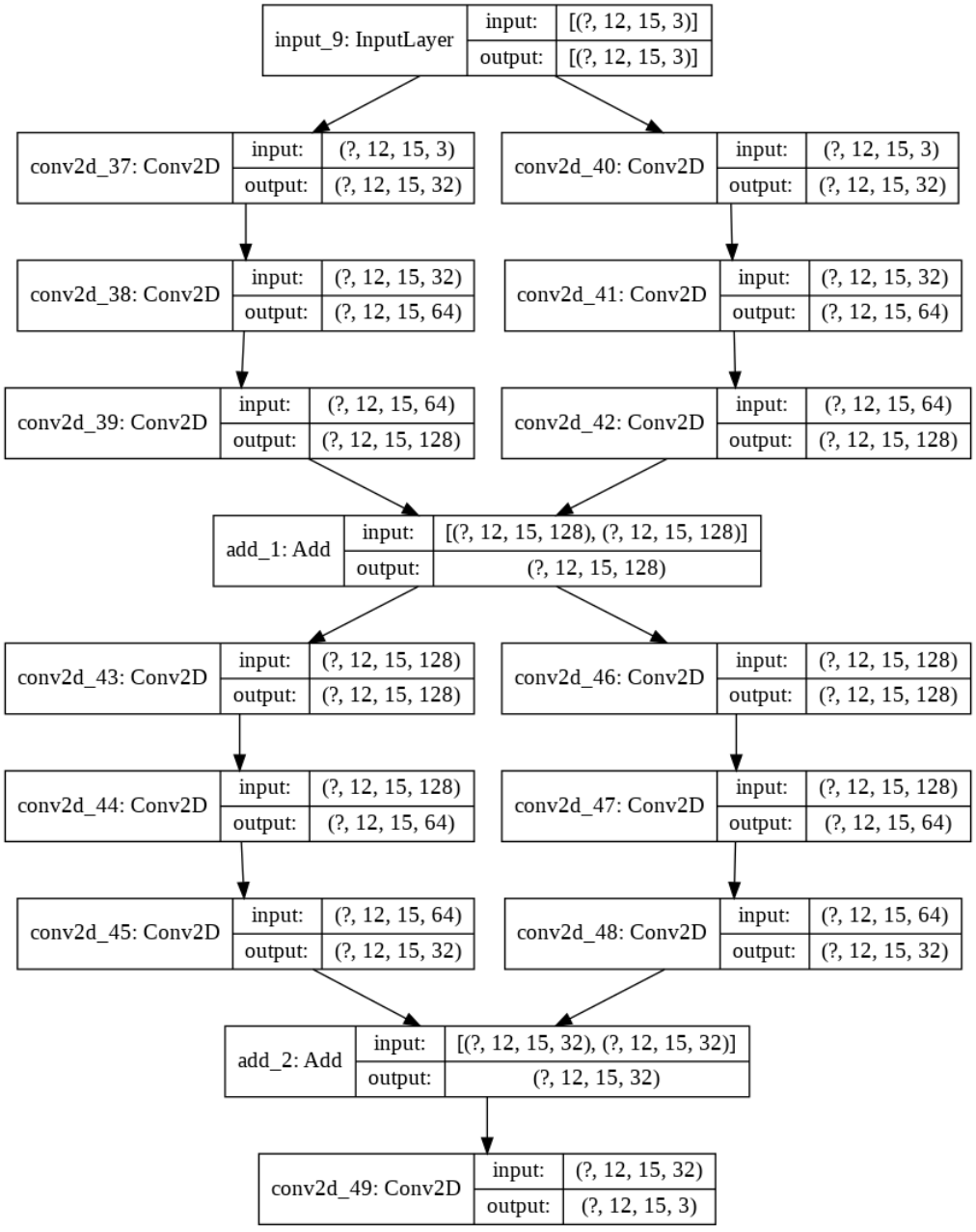


Figure B.15.: CAE 8

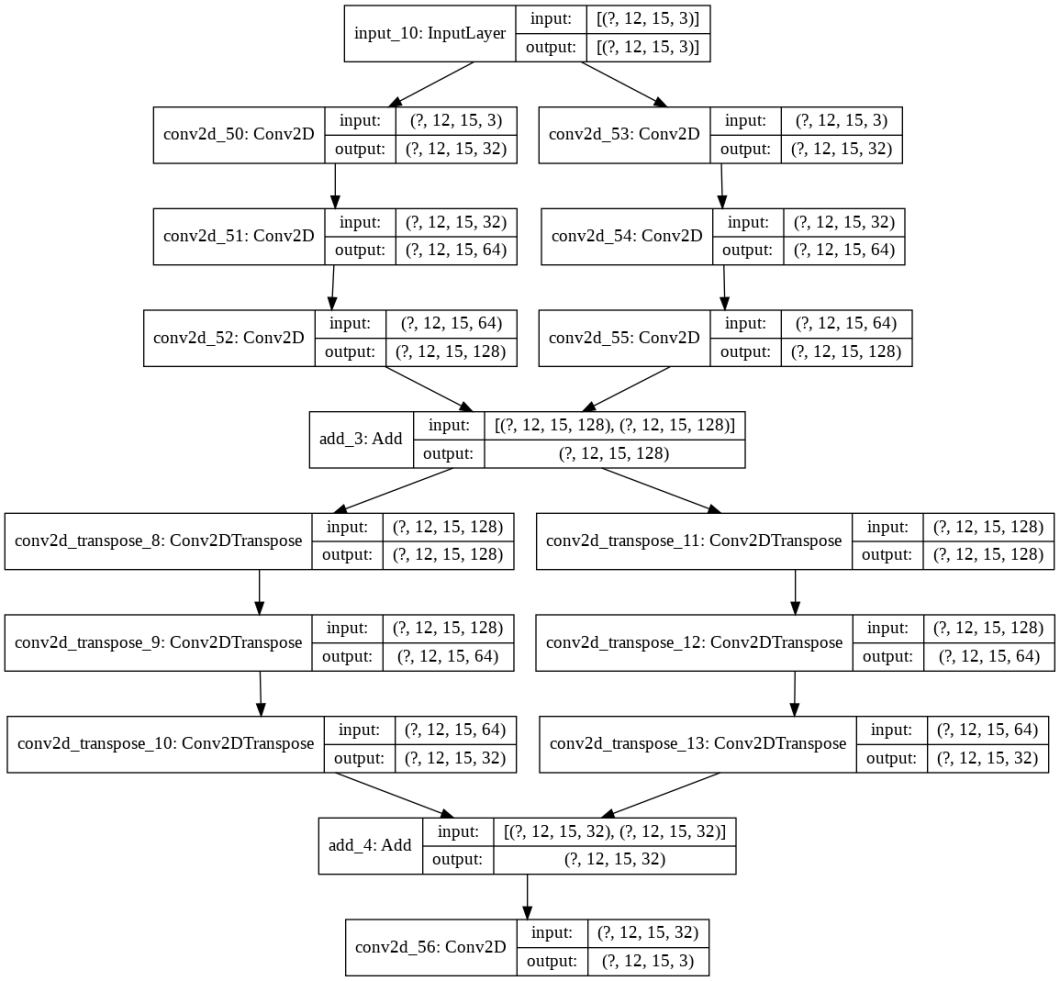


Figure B.16.: CAE 8.5

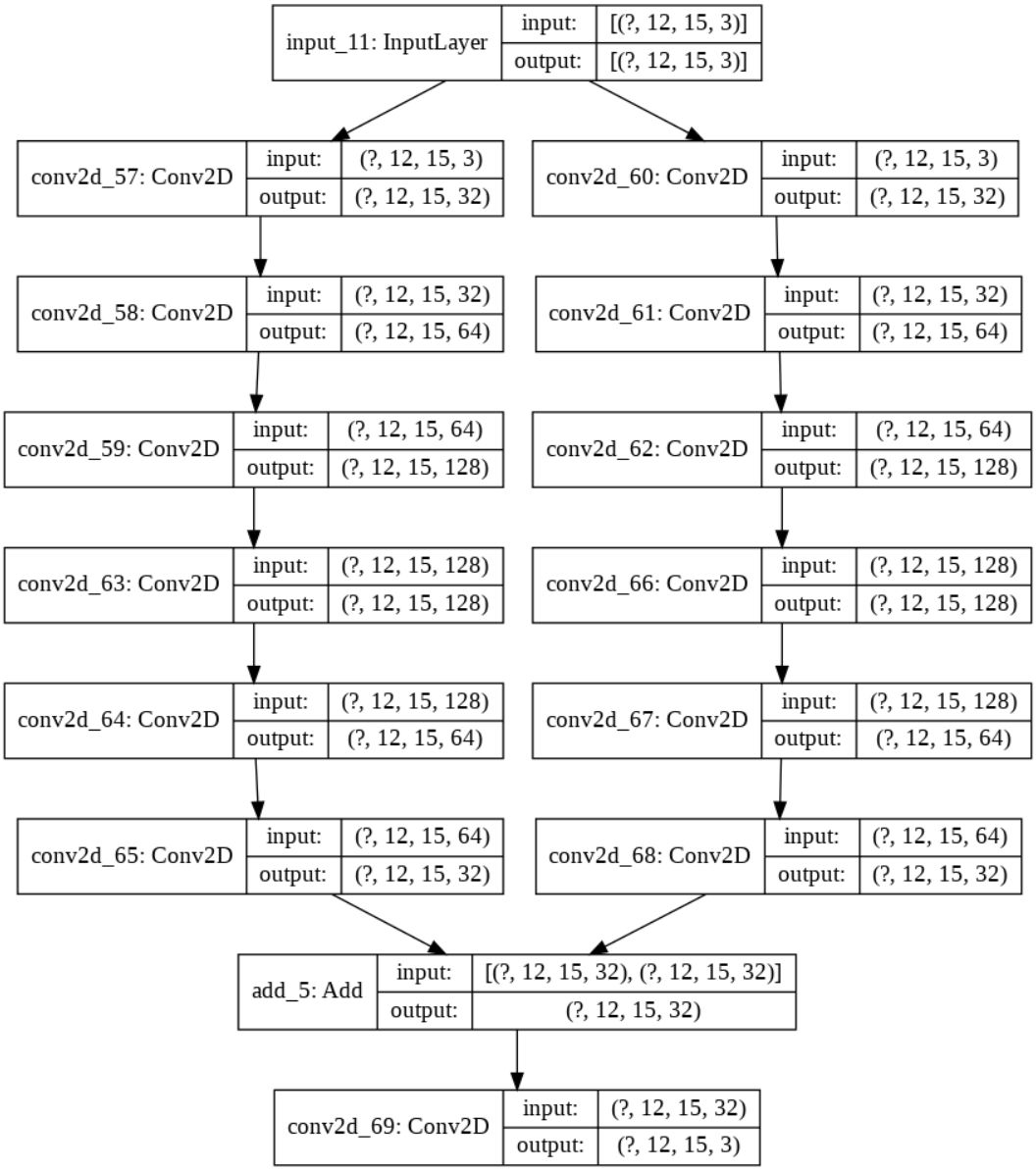


Figure B.17.: CAE 8.6

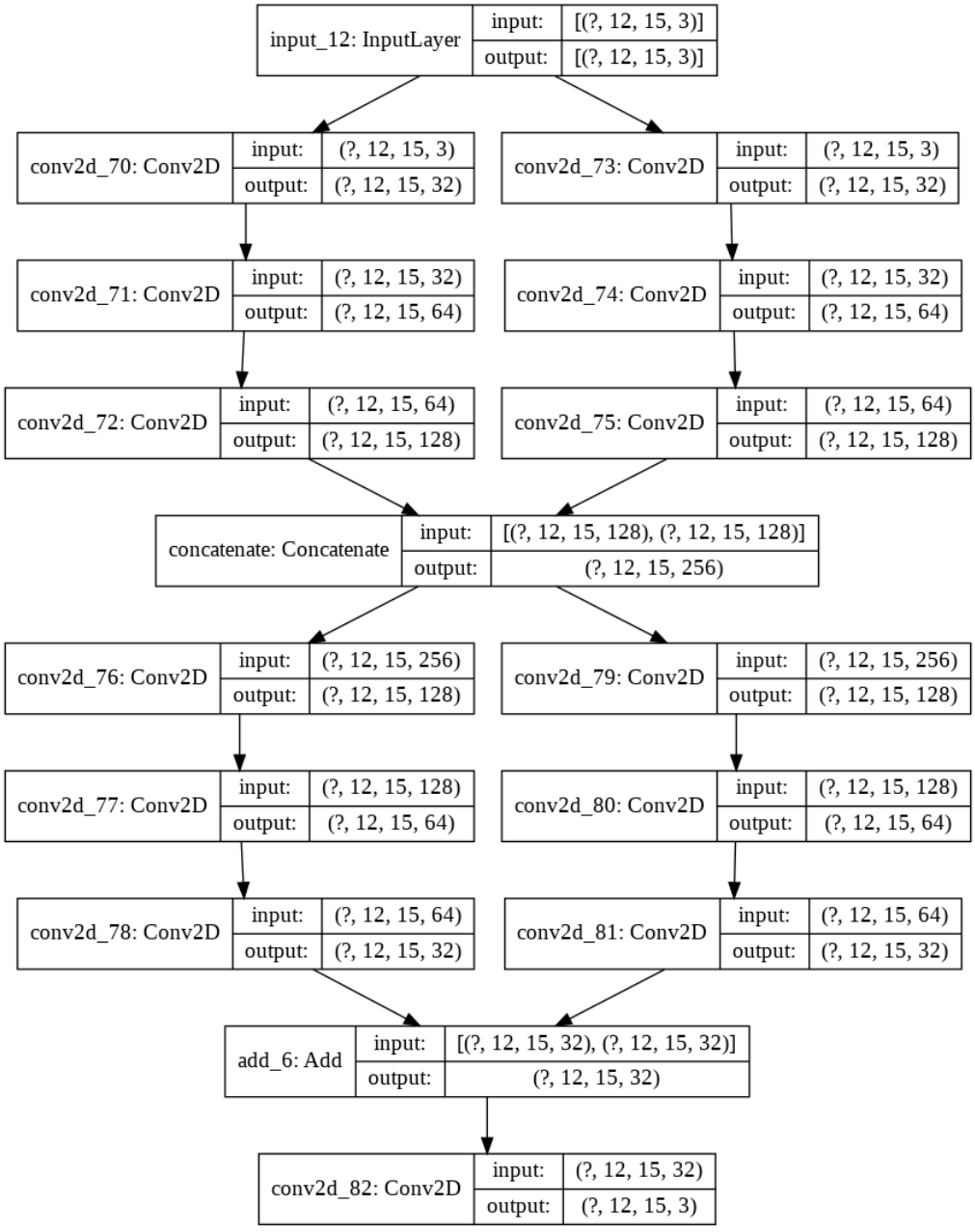


Figure B.18.: CAE 9

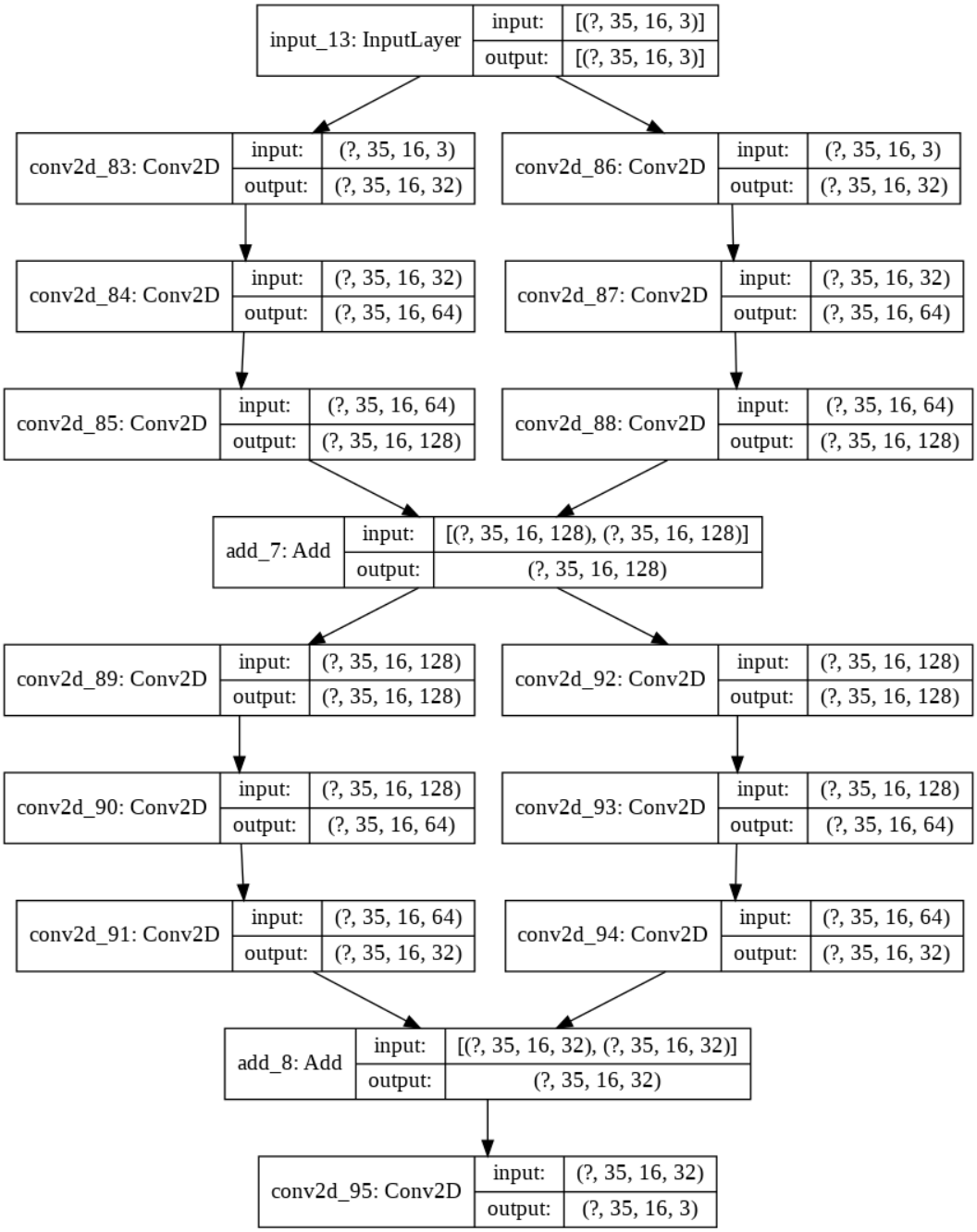


Figure B.19.: CAE 10

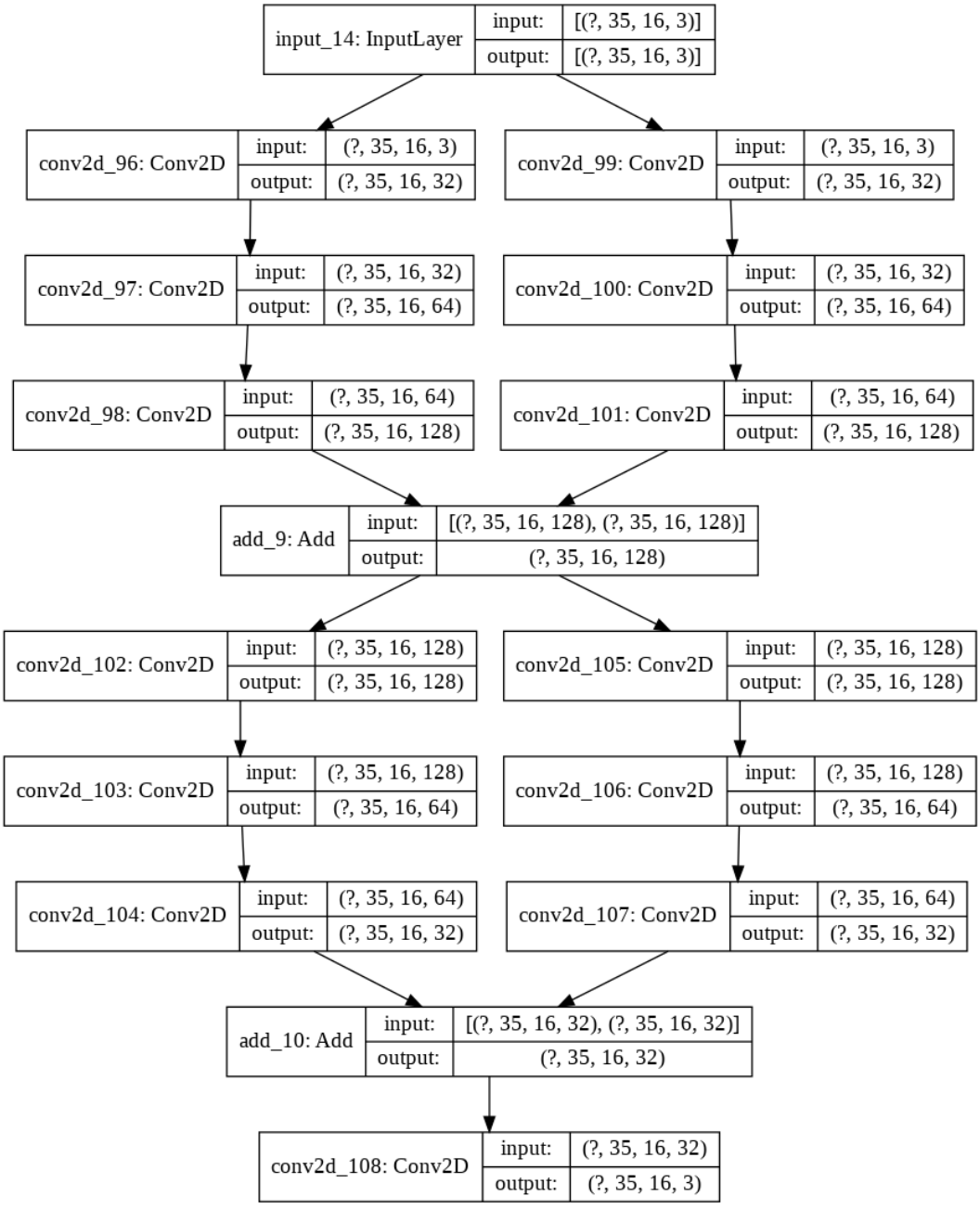


Figure B.20.: CAE 11

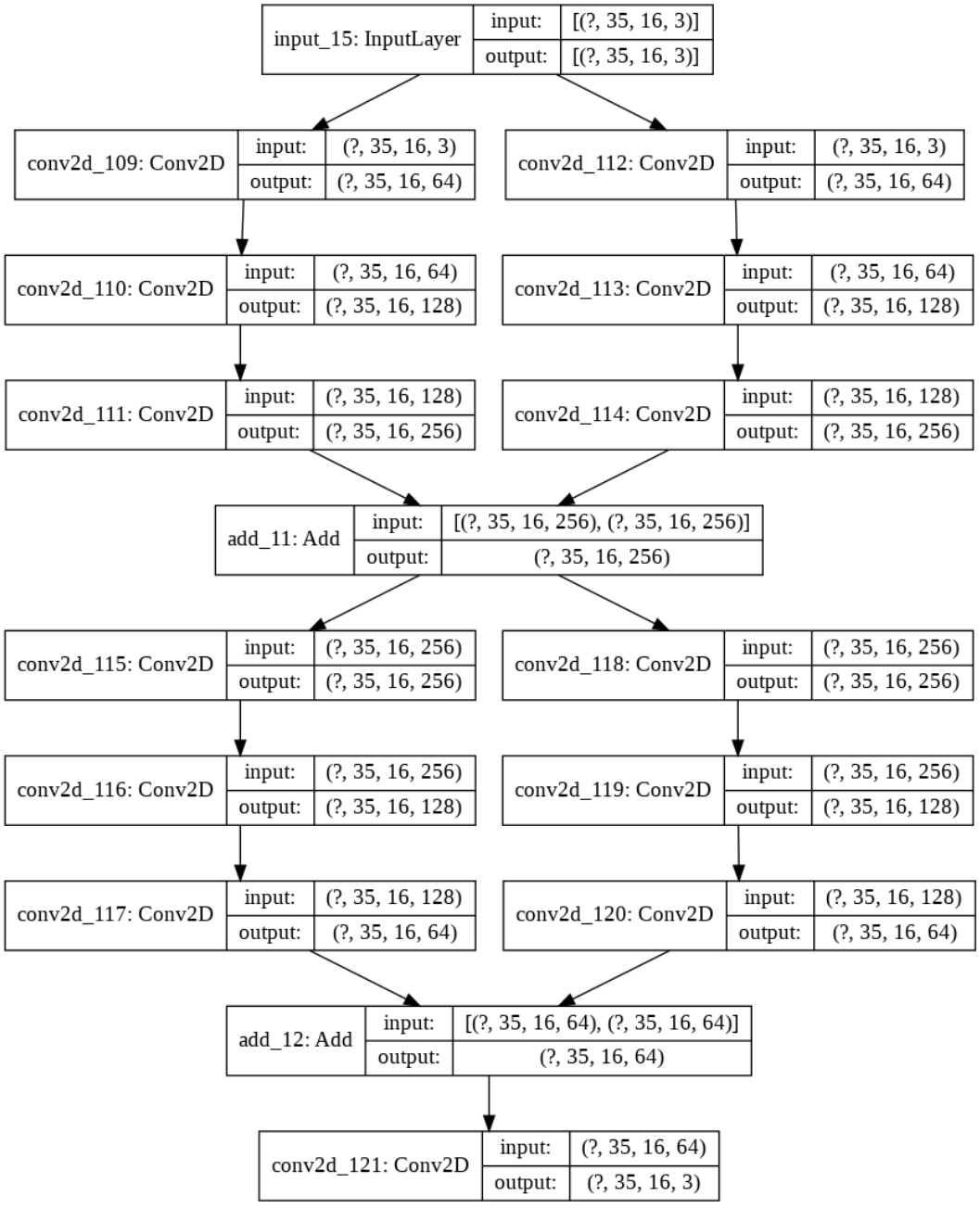


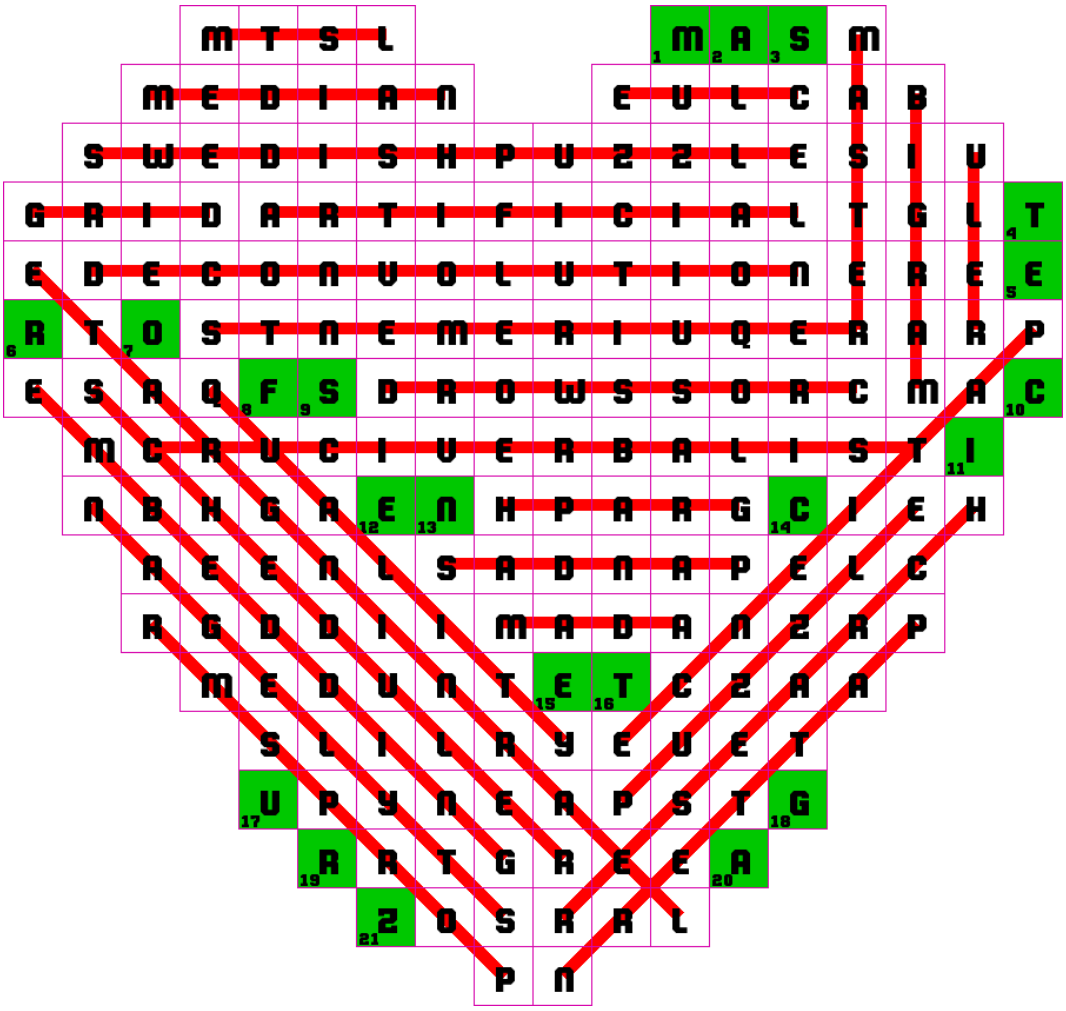
Figure B.21.: CAE 12

Appendix C.

Solution for Thank You Puzzle

"The nice thing about doing a crossword puzzle is, you know there is a solution."

- Stephen Sondheim


MTSL
MEDIAN
SWEDISHPUZZLE
GRIDARTIFICIAL
EDECONVOLUTION
RTOSTNEMERIUQERARP
ESAQFSDROWSSORCMAC
MCRUCIVEARBALIST
NBHGAENHPARGCIEH
AEENLSADNAPELC
RGDDIIMADANZRP
MEDUNTETCZAA
SLILRYEUET
UPYNEAPSTG
RRTGREEA
ZOSRRL
PN

Bibliography

[1] *1-2-3 Word Search Maker*. URL: https://www.wordsearchmaker.com/ (visited on 03/22/2020) (cit. on p. 22).

[2] Martín Abadi et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems". In: (2015) (cit. on p. 60).

[3] Aoife Aherne and Carl Vogel. "Crossing Wordnet with Crosswords, Netting Enhanced Automatic Crossword Generation". In: (Mar. 22, 2020) (cit. on p. 25).

[4] Aoife Aherne and Carl Vogel. "Wordnet Enhanced Automatic Crossword Generation". In: (2006) (cit. on pp. 25, 26).

[5] Bagus Alghani. "THE USE OF CROSSWORD PUZZLE GAME AND CLUSTERINGTECHNIQUE ON EXTROVERT AND INTROVERT STUDENTS'VOCABULARY SIZE". In: 2017 (cit. on p. 20).

[6] Bagus Alghani, Cucu Sutarsyah, and Ari Nurweni. "THE USE OF CROSSWORD PUZZLE GAME AND CLUSTERING TECHNIQUE ON VOCABULARY SIZE". In: 2017 (cit. on p. 20).

[7] Rashid Ali, Brian Black, and Jeffrey Spiro. *Investigation of Dynamically Generated Crossword Puzzles*. May 1, 2003 (cit. on p. 26).

[8] Angel C. de Dios. *A Word Search in DepEd's K to 12 Learning Module*. URL: https://www.philippinesbasiceducation.us/2015/12/a-word-search-in-depeds-k-to-12.html (visited on 03/22/2020) (cit. on p. 15).

[9] Gerhard Arminger and Daniel Enache. “Statistical Models and Artificial Neural Networks”. In: *Data Analysis and Information Systems*. Ed. by Hans-Hermann Bock and Wolfgang Polasek. Studies in Classification, Data Analysis, and Knowledge Organization. Berlin, Heidelberg: Springer, 1996, pp. 243–260 (cit. on p. 83).

[10] *Armored Penguin*. URL: https://www.armoredpenguin.com/wordsearch/ (visited on 03/22/2020) (cit. on p. 22).

[11] Bhavna Arora and NS Kumar. “Automatic Keyword Extraction and Crossword Generation Tool for Indian Languages: SEEKH”. In: *2019 IEEE Tenth International Conference on Technology for Education (T4E)*. 2019 IEEE Tenth International Conference on Technology for Education (T4E). Dec. 2019, pp. 272–273 (cit. on p. 24).

[12] Ika Fitria Astutik. “THE USE OF WORD SEARCH GAME TO DEVELOP STUDENTS’ MASTERY OF VOCABULARY OF THE SEVENTH YEAR STUDENTS OF MTS TARQIYATUL HIMMAH IN THE ACADEMIC YEAR OF 2013/2014”. other. IAIN Salatiga, Oct. 2014. 65 pp. (cit. on p. 24).

[13] C M Bailey, C T Hsu, and S E DiCarlo. “Educational Puzzles for Understanding Gastrointestinal Physiology.” In: *Advances in Physiology Education* 276.6 (June 1, 1999), S1 (cit. on p. 24).

[14] Katharina Barbe. “Propaganda in the Trivial: Puzzles in the Women’s Section of the Völkischer Beobachter”. In: *Discourse & Communication* 2.2 (May 1, 2008), pp. 115–141 (cit. on p. 21).

[15] Adam Beacham et al. “Constraint Programming Lessons Learned from Crossword Puzzles”. In: *Advances in Artificial Intelligence*. Ed. by Eleni Stroulia and Stan Matwin. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2001, pp. 78–87 (cit. on p. 25).

[16] *Benefits of Active Recall*. URL: https://www.educationquizzes.com/us/tutors/ (visited on 03/22/2020) (cit. on p. 15).

[17] H. Berghel. “Crossword Compilation with Horn Clauses”. In: *The Computer Journal* 30.2 (Jan. 1, 1987), pp. 183–188 (cit. on p. 25).

[18] H. Berghel and R. Rankin. “A Proposed Standard for Measuring Crossword Compilation Efficiency”. In: *The Computer Journal* 33.2 (Jan. 1, 1990), pp. 181–184 (cit. on p. 30).

[19] H. Berghel and C. Yi. “Crossword Compiler-Compilation”. In: *The Computer Journal* 32.3 (Jan. 1, 1989), pp. 276–280 (cit. on p. 25).

[20] David C. Berry and Michael G. Miller. “Crossword Puzzles as a Tool to Enhance Athletic Training Student Learning: Part 2”. In: *International Journal of Athletic Therapy and Training* 13.1 (Jan. 1, 2008), pp. 32–34 (cit. on p. 16).

[21] David C. Berry and Michael G. Miller. “Crossword Puzzles as a Tool to Enhance Athletic Training Student Learning: Part I”. In: *Athletic Therapy Today* 13.1 (2008), pp. 29–31 (cit. on p. 16).

[22] Ekaba Bisong. “Google Colaboratory”. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Ed. by Ekaba Bisong. Berkeley, CA: Apress, 2019, pp. 59–64 (cit. on p. 60).

[23] Douglas Bonomo, Adrian Lauf, and Roman Yampolskiy. “A Crossword Puzzle Generator Using Genetic Algorithms with Wisdom of Artificial Crowds”. In: July 1, 2015, pp. 44–49 (cit. on p. 24).

[24] Adi Botea. “Crossword Grid Composition with a Hierarchical CSP Encoding”. In: Sept. 1, 2007 (cit. on p. 25).

[25] George L. Breeler. “Integrated Crossword and Circle-a-Word Puzzle”. U.S. pat. 6308954B1. George L Breeler. Oct. 30, 2001 (cit. on p. 21).

[26] Christine Broome. “Cross-Word Puzzles for Spanish Classes”. In: *The Modern Language Journal* 9.7 (1925), pp. 431–436 (cit. on p. 17).

[27] Amy Buttner. *Activities, Games, Assessment Strategies, and Rubrics For The Foreign Language Classroom*. Routledge, Oct. 11, 2013. 206 pp. (cit. on p. 17).

[28] David J. C and Mackay Jeremy Thorpe. *Counting Crosswords*. 2006 (cit. on p. 27).

[29] Kai Kin Michael Chan. "Word Clue Puzzle Game". U.S. pat. 20110084446A1. Kai Kin Michael Chan. Apr. 14, 2011 (cit. on p. 21).

[30] Haradhan Chel, Deepak Mylavarapu, and Deepak Sharma. "A Novel Multistage Genetic Algorithm Approach for Solving Sudoku Puzzle". In: Mar. 1, 2016, pp. 808–813 (cit. on p. 28).

[31] Laura Chesy, Endang Susilawati, and Eusabinus Bunau. "THE USE OF WORD SEARCH PUZZLES TO TEACH STUDENTS' VOCABULARY MASTERY". In: *Jurnal Pendidikan dan Pembelajaran Khatulistiwa* 7.9 (Sept. 13, 2018) (cit. on p. 17).

[32] Cheryl D. Childers. "Using Crossword Puzzles as an Aid to Studying Sociological Concepts". In: *Teaching Sociology* 24.2 (1996), pp. 231–235 (cit. on p. 15).

[33] François Chollet et al. "Keras". In: (2015) (cit. on p. 60).

[34] Aaron Christensen and Gabe Emerson. "Sudoku Puzzles as a Constraint Satisfaction Problem". In: 2007 (cit. on p. 28).

[35] Russell B. Clayton, Glenn Leshner, and Anthony Almond. "The Extended iSelf: The Impact of iPhone Separation on Cognition, Emotion, and Physiology". In: *Journal of Computer-Mediated Communication* 20.2 (Mar. 1, 2015), pp. 119–135 (cit. on p. 20).

[36] *CMFC APP*. URL: http://www.cmfcapp.com/ (visited on 03/22/2020) (cit. on p. 22).

[37] Galen Collins. "Crossword Puzzles: Playing To Win or Learn?:" in: *Hospitality Education and Research Journal* (Sept. 15, 2016) (cit. on p. 18).

[38] James L. Cox, Stephen Lucci, and Tayfun Pay. “Effects of Dynamic Variable - Value Ordering Heuristics on the Search Space of Sudoku Modeled as a Constraint Satisfaction Problem”. In: *Inteligencia Artificial* 22.63 (Jan. 10, 2019), pp. 1–15 (cit. on p. 28).

[39] *CPT Crosswords*. URL: https://www.softpedia.com/get/Others/Home-Education/CPT-Crosswords.shtml (visited on 04/08/2020) (cit. on p. 22).

[40] *CrossFire*. URL: http://beekeeperlabs.com/crossfire/ (visited on 03/22/2020) (cit. on pp. 2, 22).

[41] Edward K. Crossman and Sharyn M. Crossman. “The Crossword Puzzle as a Teaching Tool”. In: *Teaching of Psychology* 10.2 (Apr. 1, 1983), pp. 98–99 (cit. on p. 16).

[42] *Crossword Compiler*. URL: https://www.crossword-compiler.com/de/ (visited on 03/22/2020) (cit. on p. 22).

[43] *Crossword Express*. URL: https://www.softpedia.com/get/Others/Home-Education/Crossword-Express-Compiler.shtml (visited on 04/08/2020) (cit. on p. 22).

[44] *Crossword Express Pro*. URL: http://www.puzzledepot.com/cwe/cwe_man.pdf (visited on 03/22/2020) (cit. on p. 22).

[45] *Crossword Hobbyist*. URL: https://CrosswordHobbyist.com (visited on 03/22/2020) (cit. on p. 22).

[46] *Crossword Labs*. URL: https://crosswordlabs.com/ (visited on 03/22/2020) (cit. on p. 22).

[47] *Crossword Solver*. URL: https://www.crosswordsolver.org/ (visited on 03/22/2020) (cit. on p. 22).

[48] *Crossword Weaver*. URL: https://www.crosswordweaver.com/index.html (visited on 03/22/2020) (cit. on p. 22).

[49] David Kriesel. *Xerox Scanners/Photocopiers Randomly Alter Numbers in Scanned Documents*. Sept. 5, 2017. URL: http://www.dkriesel.com/en/blog/2013/0802_xerox-workcentres_are_switching_written_numbers_when_scanning (visited on 07/19/2020) (cit. on p. 88).

[50] David Shulman. “How It Began”. In: () (cit. on p. 13).

[51] David Steinberg. *David Steinberg on Twitter: "For Those of You Who Are New to the Universal Crossword, Pay Is $110 for a 15x15 and $260 for a 21x21, and My Email Address for Submissions Is Amsxwords@gmail.Com." / Twitter*. URL: https://twitter.com/dsteinberg49/status/1239394672235216899 (visited on 03/22/2020) (cit. on p. 21).

[52] Tricia M Davis, Brooke Shepherd, and Tara Zwiefelhofer. “Reviewing for Exams: Do Crossword Puzzles Help in the Success of Student Learning?” In: 9.3 (2009), p. 7 (cit. on p. 15).

[53] Ruth S. Day. “Differences between Language-Bound and Stimulus-Bound Subjects in Solving Word Search Puzzles”. In: *The Journal of the Acoustical Society of America* 55.2 (Feb. 1, 1974), pp. 412–412 (cit. on p. 20).

[54] Barbara De Kegel and Mads Haahr. “Procedural Puzzle Generation: A Survey”. In: *IEEE Transactions on Games* 12.1 (Mar. 2020), pp. 21–40 (cit. on p. 30).

[55] Peter Andreasen December and Peter Andreasen. *Crosswords and Information Theory* (cit. on p. 27).

[56] Rina Dechter and Itay Meiri. “Experimental Evaluation of Preprocessing Algorithms for Constraint Satisfaction Problems”. In: *Artificial Intelligence* 68.2 (Aug. 1, 1994), pp. 211–241 (cit. on p. 33).

[57] *Die Suchsel-Maschine*. URL: http://suchsel.bastelmaschine.de (visited on 05/24/2020) (cit. on p. 22).

[58] *Discovery Education’s Puzzlemaker!* URL: http://puzzlemaker.discoveryeducation.com/WordSearchSetupForm.asp (visited on 04/08/2020) (cit. on p. 22).

[59] Charles Ditter. “Three-Dimensional Word-Search Puzzle and Methods for Making and Playing the Three-Dimensional Word-Search Puzzle”. U.S. pat. 20050253335A1. Ditter Charles W. Nov. 17, 2005 (cit. on p. 21).

[60] Dr. John M. Weiss. “Genetic Algorithms and Sudoku”. In: () (cit. on p. 28).

[61] Laura B. Edge. *From Jazz Babies to Generation Next: The History of the American Teenager*. Twenty-First Century Books, Mar. 1, 2011. 116 pp. (cit. on p. 13).

[62] *Education.Com Crossword Puzzle Maker*. URL: https://www.education.com/worksheet-generator/reading/crossword-puzzle/ (visited on 03/22/2020) (cit. on p. 22).

[63] Miles Efron. *Shannon Meets Shortz: A Probabilistic Model of Crossword Puzzle Difficulty*. Apr. 1, 2008 (cit. on p. 27).

[64] Emrah Aydemir and Zulfu Genc. "An Online Dynamic Cross-Puzzle Generation Algorithm and Its Performance". In: () (cit. on p. 23).

[65] Jakob Engel et al. "On Computer Integrated Rationalized Crossword Puzzle Manufacturing". In: June 4, 2012, pp. 131–141 (cit. on p. 24).

[66] Marco Ernandes, Giovanni Angelini, and Marco Gori. "WebCrow: A Web-Based System for Crossword Solving". In: *AAAI*. 2005 (cit. on p. 26).

[67] Luis Espinosa-Anke et al. "DefExt: A Semi Supervised Definition Extraction Tool". In: *Language Resources and Evaluation* (May 24, 2016) (cit. on p. 26).

[68] Jennifer Esteche et al. "Automatic Definition Extraction and Crossword Generation From Spanish News Text". In: *CLEI Electronic Journal* 20 (Aug. 1, 2017), p. 6 (cit. on p. 26).

[69] Bertram Felgenhauer and Frazer Jarvis*. *Mathematics of Sudoku I*. URL: https://docplayer.net/21086991-Mathematics-of-sudoku-i.html (visited on 03/22/2020) (cit. on p. 29).

[70] Bertram Felgenhauer and Frazer Jarvis. "Enumerating Possible Sudoku Grids". In: (July 20, 2005) (cit. on p. 29).

[71] Kevin K. Ferland. "Record Crossword Puzzles". In: *The American Mathematical Monthly* 121.6 (June 1, 2014), pp. 534–536 (cit. on p. 15).

[72] Sue Franklin, Mary Peat, and Alison Lewis. “Non-Traditional Interventions to Stimulate Discussion: The Use of Games and Puzzles”. In: *Journal of Biological Education* 37.2 (Mar. 1, 2003), pp. 79–84 (cit. on p. 16).

[73] Georgiana R. Frayer-Luna. *Ho'opilipili 'Olelo: Hawaiian Language Crossword Puzzles, Word Search Puzzles, and Crossword Dictionary*. University of Hawaii Press, 2000. 151 pp. (cit. on p. 18).

[74] Tracy Fullerton. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games, Third Edition*. CRC Press, Mar. 5, 2014. 528 pp. (cit. on p. 2).

[75] Ibrahim Musa Garba. “Morphological and Syntactic Meaning: An Interactive Crossword Puzzle Approach”. In: 2016 (cit. on p. 18).

[76] Nabil N. Ghaly. “Electronic Word Puzzle”. U.S. pat. 7618313B2. Ghaly Nabil N. Nov. 17, 2009 (cit. on p. 21).

[77] Gheorghita Ghinea and Oluwakemi Ademoye. “Olfactory Media Impact on Task Performance: The Case of a Word Search Game”. In: *2015 International Conference on Interactive Mobile Communication Technologies and Learning (IMCL)*. 2015 International Conference on Interactive Mobile Communication Technologies and Learning (IMCL). Nov. 2015, pp. 296–300 (cit. on p. 20).

[78] Matthew L. Ginsberg et al. “Search Lessons Learned from Crossword Puzzles”. In: *AAAI*. 1990 (cit. on p. 23).

[79] Jeremy Lee Graybill. “Integrated Word-Search, Word-Link, Trivia Puzzle and Word-Scramble”. U.S. pat. 20070267815A1. Jeremy Lee Graybill. Nov. 22, 2007 (cit. on p. 21).

[80] Geoff Harris, John Forster, and Richard Rankin. “Basic Blocks in Unconstrained Crossword Puzzles”. In: *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice*. SAC '93. Indianapolis, Indiana, USA: Association for Computing Machinery, Mar. 1, 1993, pp. 257–262 (cit. on p. 23).

[81] Geoff Harris, John Forster, and Richard Rankin. *The Ripple Effect*. Feb. 1, 1994 (cit. on p. 23).

[82] Terry L. Helser. "Terminology: Four Puzzles from One Word-search". In: *Journal of Chemical Education* 80.4 (Apr. 1, 2003), p. 414 (cit. on p. 68).

[83] Nicholas Henriquez and Liz Maynes-Aminzade. "Reintroducing The New Yorker's Cryptic Crossword". In: (Nov. 26, 2019) (cit. on p. 30).

[84] Hiroshi Higashida. "The Role of Computers in Puzzle World". In: *Proceedings of the 2011 Second International Conference on Culture and Computing*. CULTURE-COMPUTING '11. USA: IEEE Computer Society, Oct. 20, 2011, pp. 141–142 (cit. on p. 14).

[85] Vossoughi Hossein and Zargar Marzieh. "USING WORD-SEARCH-PUZZLE GAMES FOR IMPROVING VOCABULARY KNOWLEDGE OF IRANIAN EFL LEARNERS". In: 1.1 (Jan. 1, 2009), pp. 79–85 (cit. on p. 19).

[86] *How to Make a Crossword Puzzle: Detailed Instructions for Beginners*. URL: https://crosswordhobbyist.com/how-to-make-a-crossword-puzzle (visited on 03/25/2020) (cit. on p. 1).

[87] *How to Make Crossword Puzzles*. URL: https://www.wikihow.com/Make-Crossword-Puzzles (visited on 03/24/2020) (cit. on p. 14).

[88] Ryan Hughes and Roman Yampolskiy. "Solving Sudoku Puzzles with Wisdom of Artificial Crowds". In: *International Journal of Intelligent Games & Simulation* 7 (Jan. 1, 2013), p. 6 (cit. on p. 29).

[89] Hui-Chun Hung and Shelley Shwu-Ching Young. "Constructing the Game-Based Learning Environment on Handheld Devices to Facilitate English Vocabulary Building". In: *Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007)*. Seventh IEEE International Conference

on Advanced Learning Technologies (ICALT 2007). July 2007, pp. 348–350 (cit. on p. 18).

[90] J. D. Hunter. “Matplotlib: A 2D Graphics Environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95 (cit. on p. 67).

[91] Carlos Mario Zapata Jaramillo, Bell Manrique Losada, and Michael J. Fekula. “Designing and Solving Crossword Puzzles: Examining Efficacy in a Classroom Exercise”. In: 2012 (cit. on p. 19).

[92] Jessie Chin et al. *Age Differences in Information Foraging: Search and Switch in Word Search Puzzles* (cit. on p. 20).

[93] Allan Jones. “Let’s Make Learning Fun Too. Using Crossword Compiler Version 6.0”. In: *Bioscience Education* 1.1 (Jan. 1, 2003), pp. 1–7 (cit. on p. 18).

[94] Jewel O. Loud Jr. “Word Puzzle and Game”. U.S. pat. 5813672A. Loud, Jr.; Jewel O. Sept. 29, 1998 (cit. on p. 21).

[95] Tero Karras, Samuli Laine, and Timo Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks”. In: (Mar. 29, 2019) (cit. on p. 84).

[96] Greg A. Keim et al. “Proverb: The Probabilistic Cruciverbalist”. In: *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*. AAAI ’99/IAAI ’99. Orlando, Florida, USA: American Association for Artificial Intelligence, July 18, 1999, pp. 710–717 (cit. on p. 26).

[97] Keiran King. *The New York Times Crossword – Keiranking.Com*. URL: http://www.keiranking.com/blog/2018/nyt-crossword-debut/ (visited on 03/22/2020) (cit. on p. 23).

[98] Jay P. Kunz. “Creating Word Search Puzzles with a Pedagogical Purpose”. In: *Die Unterrichtspraxis / Teaching German* 35.2 (2002), pp. 148–153 (cit. on p. 17).

[99] David Kwong. *How to Create a Crossword Puzzle | WIRED*. URL: https://www.youtube.com/watch?v=aAqQnXHd7qk (visited on 03/24/2020) (cit. on p. 14).

[100] Kyle Williams. *Using a Genetic Algorithm to Solve Crossword Puzzles*. (Cit. on p. 24).

[101] Natan Last. "The Hidden Bigotry of Crosswords". In: *The Atlantic. Culture* (Mar. 18, 2020) (cit. on p. 13).

[102] Lazy Programmer. *All Data Is the Same (in Machine Learning)*. URL: https://www.youtube.com/watch?v=GJWNyRpvU-M (visited on 05/25/2020) (cit. on p. 61).

[103] Leonard Joseph Kemp. "Educational Puzzle". U.S. pat. 2544961A. Kemp Leonard Joseph. Mar. 13, 1951 (cit. on p. 21).

[104] Tin-Chun Lin and Steven M. Dunphy. "Using the Crossword Puzzle Exercise in Introductory Microeconomics to Accelerate Business Student Learning". In: *Journal of Education for Business* 88.2 (Jan. 1, 2013), pp. 88–93 (cit. on p. 15).

[105] James Little. *WHAT'S A NINE LEITER WORD FOR "A TYPE OF WORD PUZZLE„?L* (cit. on p. 15).

[106] Michael L. Littman, Greg A. Keim, and Noam Shazeer. "A Probabilistic Approach to Solving Crossword Puzzles". In: *Artificial Intelligence* 134.1 (Jan. 1, 2002), pp. 23–55 (cit. on p. 26).

[107] Michael L. Littman, Greg A. Keim, and Noam M. Shazeer. "Solving Crosswords with PROVERB". In: *AAAI/IAAI*. 1999 (cit. on p. 26).

[108] Inês Lynce and Joël Ouaknine. "Sudoku as a SAT Problem". In: *ISAIM*. 2006 (cit. on p. 29).

[109] Thomas Manzini, Simon Ellis, and James Hendler. "A Play on Words: Using Cognitive Computing as a Basis for AI Solvers in Word Puzzles". In: *Journal of Artificial General Intelligence* 6 (May 31, 2015) (cit. on p. 26).

[110] Matt Gaffney. *How Crossword Puzzles Are Really Made*. Sept. 10, 2014. URL: https://www.mentalfloss.com/article/58828/how-crossword-puzzles-are-made (visited on 03/24/2020) (cit. on p. 14).

[111] Lawrence J. Mazlack. "Computer Construction of Crossword Puzzles Using Precedence Relationships". In: *Artificial Intelligence* 7.1 (Mar. 1, 1976), pp. 1–19 (cit. on p. 23).

[112] Mary E. McElroy and Fabián A. Samaniego. "Package Them In Puzzles: Vocabulary, Culture, Conjugations". In: *Foreign Language Annals* 14.3 (1981), pp. 217–220 (cit. on p. 18).

[113] Gary Meehan and Peter Gray. "Constructing Crossword Grids: Use of Heuristics vs Constraints". In: *In: Proceedings of Expert Systems 97: Research and Development in Expert Systems XIV, SGES*. Publications, 1997, pp. 159–174 (cit. on p. 23).

[114] Mitchell Melanie. "An Introduction to Genetic Algorithms". In: (), p. 162 (cit. on p. 85).

[115] Warren Merkel. "The Potential of Crossword Puzzles in Aiding English Language Learners". In: *TESOL Journal* 7.4 (2016), pp. 898–920 (cit. on p. 18).

[116] Michael Callaghan. *A Brief Guide to the Construction of Cryptic Crossword Clues* (cit. on p. 14).

[117] Mike Vuolo. *How a Crossword Puzzle Gets Made*. URL: https://www.youtube.com/watch?v=UVcypLJiKSI (visited on 03/24/2020) (cit. on p. 14).

[118] Anthony Milton and Cesar Ortega-Sanchez. "Development and Analysis of Genetic Algorithms: Sudoku Case Study". In: *TENCON 2012 IEEE Region 10 Conference*. TENCON 2012 IEEE Region 10 Conference. Nov. 2012, pp. 1–6 (cit. on pp. 28, 29).

[119] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: (Nov. 6, 2014) (cit. on p. 84).

[120] Melanie Mitchell. *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1996 (cit. on p. 85).

[121] Alberto Moraglio, Julian Togelius, and Simon Lucas. “Product Geometric Crossover for the Sudoku Puzzle”. In: Jan. 1, 2006, pp. 470–476 (cit. on p. 29).

[122] Vusi Vincent Mshayisa. “Students’ Perceptions of Plickers and Crossword Puzzles in Undergraduate Studies”. In: *Journal of Food Science Education* n/a.n/a () (cit. on p. 19).

[123] S. Shaun Murphree. “Sudoku as a Constraint Satisfaction Problem”. In: 2012 (cit. on p. 29).

[124] *My Worksheet Maker*. URL: https://myworksheetmaker.com (visited on 03/22/2020) (cit. on p. 22).

[125] David L. Nichols. “Word Puzzle Game”. U.S. pat. 4215864A. Nichols David L. Aug. 5, 1980 (cit. on p. 21).

[126] Adelheid A. M. Nicol. “A Highly Interactive Application of Self-Generated Crosswords in the Classroom”. In: *College Teaching* 68.1 (Jan. 2, 2020), pp. 3–4 (cit. on p. 19).

[127] Stavroula Ntoa et al. “A-Cross: An Accessible Crossword Puzzle for Visually Impaired Users”. In: *Universal Access in Human-Computer Interaction. Users Diversity*. Ed. by Constantine Stephanidis. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 342–351 (cit. on p. 30).

[128] Wiwat Orawiwatnakul. “Crossword Puzzles as a Learning Tool for Vocabulary Development.” In: 2013 (cit. on p. 18).

[129] F. Pedregosa et al. “Scikit-Learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830 (cit. on p. 60).

[130] Laurent Perron and Vincent Furnon. *OR-Tools*. Version 7.2. Google. July 19, 2019. URL: https://developers.google.com/optimization/ (cit. on pp. 46, 55).

[131] E. Pershits and R. Stansifer. “Solving Diagramless Crossword Puzzles”. In: *Proceedings Sixth International Conference on Tools with Artificial Intelligence. TAI 94*. Proceedings Sixth International Conference on Tools with Artificial Intelligence. TAI 94. Nov. 1994, pp. 4–10 (cit. on p. 27).

[132] *Phil.* URL: http://www.keiranking.com/blog/2017/phil/ (visited on 03/22/2020) (cit. on p. 22).

[133] Balazs Pinter et al. "Automated Word Puzzle Generation via Topic Dictionaries". In: (June 2, 2012) (cit. on p. 26).

[134] Balázs Pintér et al. *2 Automated Word Puzzle Generation via Topic Dictionaries* (cit. on p. 26).

[135] Aaron Port and Roman Yampolskiy. "Using a GA and Wisdom of Artificial Crowds to Solve Solitaire Battleship Puzzles". In: July 1, 2012, pp. 25–29 (cit. on p. 30).

[136] Leo Postman, William O. Jenkins, and Dorothy L. Postman. "An Experimental Comparison of Active Recall and Recognition". In: *The American Journal of Psychology* 61.4 (1948), pp. 511–519 (cit. on p. 20).

[137] Titus D. M. Purdin and Geoff Harris. "A Genetic-Algorithm Approach to Solving Crossword Puzzles". In: *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice*. SAC '93. Indianapolis, Indiana, USA: Association for Computing Machinery, Mar. 1, 1993, pp. 263–270 (cit. on p. 30).

[138] *Puzzle Maker.* URL: https://www.puzzle-maker.com/CW (visited on 03/22/2020) (cit. on p. 22).

[139] Bali Ranaivo-Malancon et al. "Automatic Generation of Fill-in Clues and Answers from Raw Texts for Crosswords". In: July 1, 2013, pp. 1–5 (cit. on p. 25).

[140] Peter H. Rehm. "Method of Constructing Crossword Puzzles by Computer". U.S. pat. 5667438A. Rehm; Peter H. Sept. 16, 1997 (cit. on p. 21).

[141] Peter H. Rehm and Rachel Rehm. "Questionnaire Method of Making Topic-Specific Word Puzzle Documents". U.S. pat. 7210996B2. Peter H Rehm, Rachel Rehm. May 1, 2007 (cit. on p. 21).

[142] Ria Damayanti H. “Teaching Vocabulary Trough Word Search Puzzle to The Fifth Grade Students of SDN 01 NgaglikBlitarIn The Academic Year 2013/2014 - Institutional Repository of IAIN Tulungagung” (cit. on p. 17).

[143] Leonardo Rigutini et al. “A Fully Automatic Crossword Generator”. In: *2008 Seventh International Conference on Machine Learning and Applications*. 2008 Seventh International Conference on Machine Learning and Applications. Dec. 2008, pp. 362–367 (cit. on p. 25).

[144] Leonardo Rigutini et al. “Automatic Generation of Crossword Puzzles”. In: *International Journal on Artificial Intelligence Tools (IJAIT)* 21 (June 1, 2012), p. 22 (cit. on p. 25).

[145] Mitchell Robert. “Cross-Word Puzzle”. U.S. pat. 2050498A. Mitchell Robert. Aug. 11, 1936 (cit. on p. 21).

[146] Rohit Iyer, Amrish Jhaveri, and Krutika Parab. “A Review of Sudoku Solving Using Patterns”. In: () (cit. on p. 29).

[147] N. Samaras and K. Stergiou. “Binary Encodings of Non-Binary Constraint Satisfaction Problems: Algorithms and Experimental Results”. In: *Journal of Artificial Intelligence Research* 24 (Nov. 2, 2005), pp. 641–684 (cit. on p. 25).

[148] Yoppy Sazaki et al. “A Comparison Application of the Genetic and Steepest Ascent Hill Climbing Algorithm in the Preparation of the Crossword Puzzle Board”. In: *2018 12th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*. 2018 12th International Conference on Telecommunication Systems, Services, and Applications (TSSA). Oct. 2018, pp. 1–5 (cit. on p. 24).

[149] Yoppy Sazaki et al. “Application of the Steepest Ascent Hill Climbing Algorithm in the Preparation of the Crossword Puzzle Board”. In: July 1, 2018, pp. 1–6 (cit. on p. 24).

[150] John C. Schafer and Jo Behymer. “Cross Purposes: Computer-Generated Crossword Puzzles Link Popular Pastime with Technical Learning”. In: *Vocational Education Journal* 67.5 (1992), p. 36 (cit. on p. 18).

[151] William Schreiber-Stainthorp. “Crosswords”. In: *The Trinity Papers (2011 - present)* (Jan. 1, 2013) (cit. on p. 1).

[152] Jimmy Dale Schroeder. “Word Search Based Board Game with Directional Tiles”. U.S. pat. 6422561B1. Jimmy Dale Schroeder. July 23, 2002 (cit. on p. 21).

[153] Gordon Seaberg. “CrossWordSearch Puzzle Game”. U.S. pat. 20020117802A1. Seaberg Gordon Eric. Aug. 29, 2002 (cit. on p. 21).

[154] Mª Isabel Martínez Serna. “ACTIVE LEARNING: CREATING INTERACTIVE CROSSWORD PUZZLES”. In: (), p. 8 (cit. on p. 16).

[155] Samit Shah, Launa M. J. Lynch, and Lilia Z. Macias-Moriarity. “Crossword Puzzles as a Tool to Enhance Learning About Anti-Ulcer Agents”. In: *American Journal of Pharmaceutical Education* 74.7 (Sept. 1, 2010) (cit. on p. 16).

[156] Helmut Simonis. “Sudoku as a Constraint Problem”. In: 2005 (cit. on p. 29).

[157] G. W. Smith and J. B. H. du Boulay. “The Generational of Cryptic Crossword Clues”. In: *The Computer Journal* 29.3 (Jan. 1, 1986), pp. 282–284 (cit. on p. 26).

[158] Jordan B. L. Smith et al. “The CrossSong Puzzle: Developing a Logic Puzzle for Musical Thinking”. In: *Journal of New Music Research* 46.3 (July 3, 2017), pp. 213–228 (cit. on p. 30).

[159] P. D. Smith. “XENO: Computer-Assisted Compilation of Crossword Puzzles”. In: *The Computer Journal* 26.4 (Nov. 1, 1983), pp. 296–302 (cit. on p. 23).

[160] P. D. Smith and S. Y. Steen. “A Prototype Crossword Compiler”. In: *The Computer Journal* 24.2 (Jan. 1, 1981), pp. 107–111 (cit. on p. 23).

[161] *Solving a Sudoku with a Genetic Algorithm – Studio Houthaak*. URL: https://studiohouthaak.nl/solving-a-sudoku-with-a-genetic-algorithm/ (visited on 03/24/2020) (cit. on p. 28).

[162] *Solving Every Sudoku Puzzle*. URL: http://norvig.com/sudoku.html (visited on 05/24/2020) (cit. on p. 28).

[163] L. J. Spring et al. "A Proposed Benchmark for Testing Implementations of Crossword Puzzle Algorithms". In: *Proceedings of the 1992 ACM/SIGAPP Symposium on Applied Computing: Technological Challenges of the 1990's*. SAC '92. Kansas City, Missouri, USA: Association for Computing Machinery, Apr. 1, 1992, pp. 99–101 (cit. on p. 30).

[164] Sruthi Sankar. "PARALLELIZED SUDOKU SOLVING ALGORITHM USING OpenMP" (cit. on p. 28).

[165] Keith E. Stanovich and Richard F. West. "The Effect of Orthographic Structure on the Word Search Performance of Good and Poor Readers". In: *Journal of Experimental Child Psychology* 28.2 (Oct. 1, 1979), pp. 258–267 (cit. on p. 28).

[166] David Steinberg and Natan Last. "How to Make a Crossword Puzzle Part 2". In: *The New York Times. Crosswords & Games* (May 11, 2018) (cit. on p. 2).

[167] *Suchsel-Generator*. URL: https://www.suchsel.net/ (visited on 05/24/2020) (cit. on p. 22).

[168] Kozo Sugiyama, Ryo Osawa, and Seok-Hee Hong. "Puzzle Generators and Symmetric Puzzle Layout". In: *Proceedings of the 2005 Asia-Pacific Symposium on Information Visualisation - Volume 45*. APVis '05. Sydney, Australia: Australian Computer Society, Inc., Jan. 1, 2005, pp. 97–105 (cit. on p. 27).

[169] Anon Sukstrienwong and Patravadee Vongsumedh. "Software Development of Word Search Game on Smart Phones in English Vocabulary Learning". In: (2013), p. 6 (cit. on p. 17).

[170] Ben Tausig and Finn Vigeland. "How to Make a Crossword Puzzle Part 1". In: *The New York Times. Crosswords & Games* (Apr. 11, 2018) (cit. on p. 2).

[171] *Teacher's Corner Crossword Puzzle Maker*. URL: https://worksheets.theteacherscorner.net/make-your-own/crossword/ (visited on 03/22/2020) (cit. on p. 22).

[172] Pavel Tomozov. *Crossword Construction Using Constraint* (cit. on p. 25).

[173] F. Toyama et al. "Assembly of Puzzles Using a Genetic Algorithm". In: vol. 4. Feb. 1, 2002, 389–392 vol.4 (cit. on p. 27).

[174] Jay S. Walker and James Jorasch. "Electronic Word Puzzle Game". U.S. pat. 5921864A. Walker Asset Management LP. July 13, 1999 (cit. on p. 21).

[175] Michael Waskom et al. *Mwaskom/Seaborn: V0.10.1 (April 2020)*. Zenodo. Apr. 26, 2020. URL: https://zenodo.org/record/3767070 (visited on 07/19/2020) (cit. on p. 67).

[176] Robert S. Weisskirch. "An Analysis of Instructorcreated Crossword Puzzles for Student Review". In: *College Teaching* 54.1 (Jan. 1, 2006), pp. 198–201 (cit. on p. 16).

[177] J. M. Wilson. "Crossword Compilation Using Integer Programming". In: *The Computer Journal* 32.3 (Jan. 1, 1989), pp. 273–275 (cit. on p. 25).

[178] Alan Wise. "Web-Based Crossword Puzzles Support Revision:" in: *Active Learning in Higher Education* (2001) (cit. on p. 16).

[179] Alan Wise. "Web-Based Puzzle Program to Assist Students' Understanding of Research Methods:" in: *Active Learning in Higher Education* (2003) (cit. on p. 16).

[180] *Wolfram Demonstrations Project*. URL: http://demonstrations.wolfram.com/CrosswordGridMaker/ (visited on 03/22/2020) (cit. on p. 22).

[181] *Word Search Generator*. URL: https://tools.atozteacherstuff.com/word-search-maker/wordsearch.php (visited on 04/08/2020) (cit. on p. 22).

[182] *Word Search Maker*. URL: https://thewordsearch.com/maker/ (visited on 04/08/2020) (cit. on p. 22).

[183] *Word Search Maker | Education.Com*. URL: https://www.education.com/worksheet-generator/reading/word-search/ (visited on 03/22/2020) (cit. on p. 22).

[184] *XWords*. URL: https://www.xwords-generator.de/en (visited on 03/22/2020) (cit. on p. 22).

[185] Li Yong-zhuo. "Sudoku Puzzles Generating: From Easy to Evil". In: 2009 (cit. on p. 29).

[186] Li Yujian and Liu Bo. "A Normalized Levenshtein Distance Metric". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (June 2007), pp. 1091–1095 (cit. on p. 85).

[187] Giulio Zambon. "Puzzle Statistics and More Puzzles". In: *Sudoku Programming with C*. Ed. by Giulio Zambon. Berkeley, CA: Apress, 2015, pp. 211–226 (cit. on p. 29).

[188] Sydney S. Zentall, Arlene M. Hall, and David L. Lee. "Attentional Focus of Students with Hyperactivity During a Word-Search Task". In: *Journal of Abnormal Child Psychology* 26.5 (Oct. 1, 1998), pp. 335–343 (cit. on p. 20).

[189] Han Zhang et al. "StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks". In: (Aug. 4, 2017) (cit. on p. 84).