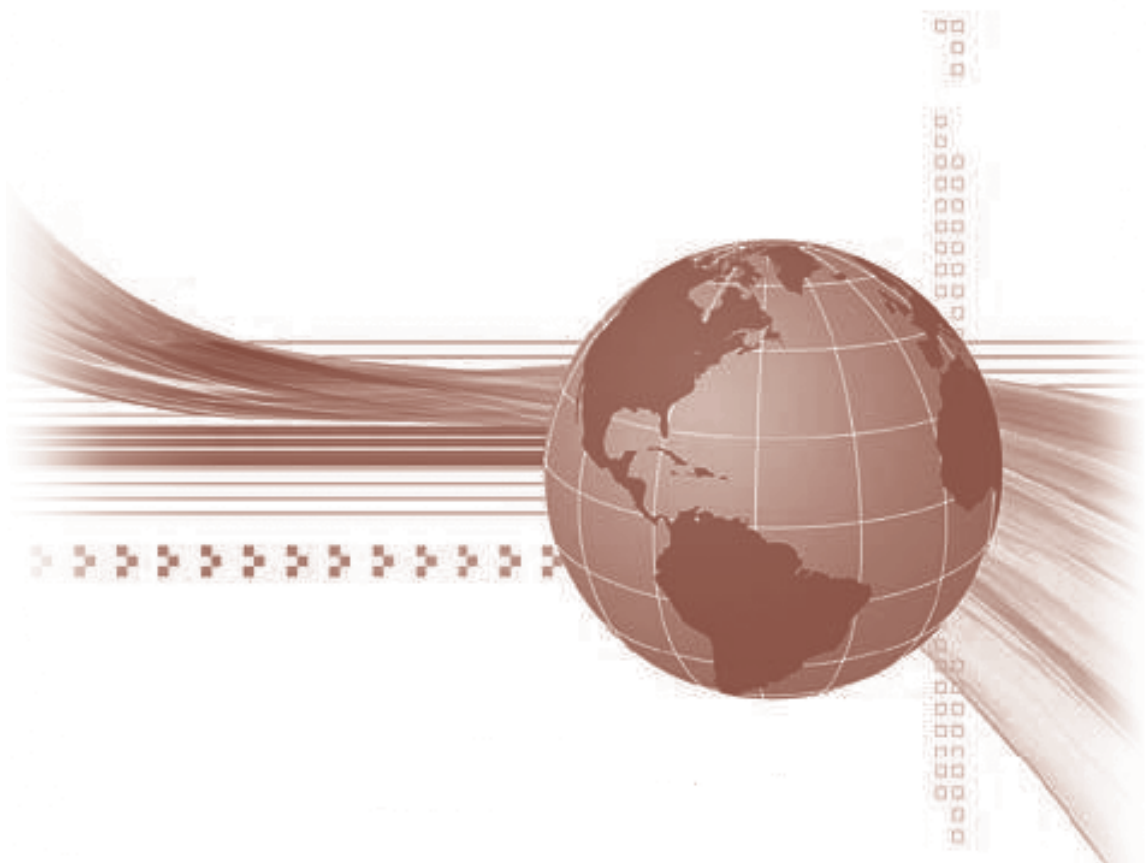
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



# INFORMATICA

1/2022

# STUDIA

## UNIVERSITATIS BABEŞ-BOLYAI
## INFORMATICA

**No. 1/2022**

**January - June**

# EDITORIAL BOARD

**YEAR Volume 67 (LXVII)** **2022**
**MONTH** **JUNE**
**ISSUE** **1**

# S T U D I A
# UNIVERSITATIS BABEȘ-BOLYAI
# INFORMATICA

**1**

**EDITORIAL OFFICE**: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

*SUMAR – CONTENTS – SOMMAIRE*

A. Bajcsi, B, Botos, P. Bájko, Z. Bodó, *Can You Guess the Title? Generating Emoji Sequences for Movies* ..... 5

I. Prăjescu, A.D. Călin, *Multiple Types of AI and Their Performance in Video Games* .... 21

B.A. Diaconu, B. Lázár-Lőrincz, *Romanian Question Answering Using Transformer Based Neural Networks* ..... 37

A. Petrescu, *Music Recommendations Based on User's Mood Using Convolutional Neural Networks*..... 45

A.R. Alexandrescu, A. Manole, *A Dynamic Approach for Railway Semantic Segmentation*..... 61

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXVII**, Number 1, 2022
DOI: 10.24193/subbi.2022.1.01

# CAN YOU GUESS THE TITLE? GENERATING EMOJI SEQUENCES FOR MOVIES

ANNA BAJCSI, BARBARA BOTOS, PÉTER BAJKÓ, AND ZALÁN BODÓ

Abstract. In the culture of the present emojis play an important role in written/typed communication, having a primary role of supplementing the words with emotional cues. While in different cultures emojis can be interpreted and thus used differently, a small set of emojis have clear meaning and strong sentiment polarity. In this work we study how to map natural language texts to emoji sequences, more precisely, we automatically assign emojis to movie subtitles/scripts. The pipeline of the proposed method is as follows: first the most relevant words are extracted from the movie subtitle, and then these are mapped to emojis. In order to perform the mapping, three methods are proposed: a lexical matching-based, a word embedding-based and a combined approach. To demonstrate the viability of the approach, we list some of the generated emojis for a randomly selected movie subset, showing also the deficiencies of the method in generating guessable sequences. Evaluation is performed via quizzes completed by human participants.

## 1. Introduction

Emojis and emoticons are commonly used to express emotions in online written communication. They are preferred tools, because in written communication mimics and tone are hard to convey, however, it is much more easily achieved by emojis.

In this paper we try to tackle a creative problem, to generate emoji sequences describing a movie. While guessing the movie title from – usually

Received by the editors: 3 November 2021.

2010 *Mathematics Subject Classification.* 68T50, 68T30.

1998 *CR Categories and Descriptors.* I.2.7 [**ARTIFICIAL INTELLIGENCE**]: Natural Language Processing – *Text analysis*; I.2.m [**ARTIFICIAL INTELLIGENCE**]: Miscellaneous.

*Key words and phrases.* natural language processing, emoji, keyword extraction, movie scripts, lexical matching, word embedding.

human-generated – emoji sequences is a popular game for movie enthusiasts[1], generating emojis describing a movie can be considered a task similar to automatic (text) summarization [19]. The main idea is to extract keywords from the movie subtitle and match them with emojis. In the present work we calculate *tf-idf* scores to obtain the most important, most representative words of the movie script. The most difficult part of the emoji sequence generation is mapping the words to emojis. We describe two main approaches in this paper: lexical matching between keywords and emoji names, as well as a word embedding-based method. We also try to improve on the results by combining these two approaches, as well as by taking into account the title of the movie and the chronological order of the keywords.

The rest of the paper is organized as follows. In Section 2 we briefly describe how emojis shape the online media today, and Section 3 presents the natural language processing problems where emojis can be useful instruments. The structure of our system generating emoji sequences from movie scripts is presented and detailed in Section 4. The experiments and the results obtained are described in Section 5, and the paper concludes with Section 6, discussing the results and specifying potential future research directions.

## 2. Emojis in online media

Emoticons are inventions of the 19th century, but the first recorded online use occurred only in 1982 [11]. The word emoticon stems from "emotion icon" [6], referring to a pictorial representation of a facial expression, gesture using characters (e.g. punctuation marks, parentheses, etc.). While in North America the horizontal representation of such faces became prevalent, e.g. `:-)`, in Japan the so-called kaomojis had been used [24], which are vertical emoticons like `(^_^)`. Emojis – meaning pictographs in Japanese (*e* = picture, *moji* = characters) [2] – appeared in the late 1990s at the NTT Docomo telecommunications company as the work of the designer Shigetaka Kurita [24]. The resemblance to the English word "emotion" or even "emoticon" is merely coincidental [25]. As pointed out in [24], the appearance of emojis in Japan could be explained by the complexity of the predecessor kaomojis.

[1]See for example the BuzzFeed movie quiz *"If You Can Identify 8/10 Of These Movies From The Emojis, You're Officially A Cinephile"* (`https://www.buzzfeed.com/hayleyrochelletillett/identify-movies-by-emojis`).

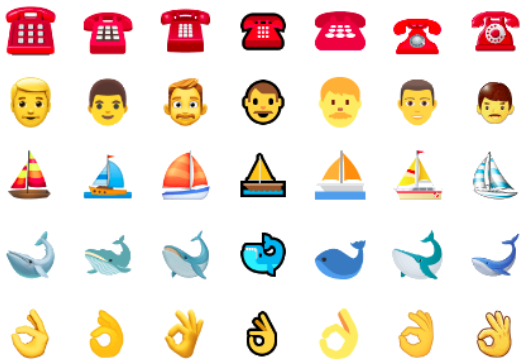


FIGURE 1. "Call me Ishmael." – the opening sentence of Herman Melville's *Moby Dick* represented as a (vertical) list of emojis in Fred Benenson's *Emoji Dick*, reproduced after [21]. The 7 columns represent different vendors: Apple, Google, Facebook, Windows, Twitter, JoyPixels, and Samsung.

The standardization of emojis took place in Unicode 6.0, containing 722 emojis, Kurita's initial set having only 176 pictograms [12]. The current Unicode 14.0 standard includes 3633 emojis.[2]

Emojis became popular worldwide only in the 2010s, by appearing in various mobile operating systems and web browsers. To emphasize their increasing popularity, we mention the first ever published book written using emojis only, appeared in 2010: Fred Benenson's *Emoji Dick*, the crowdsourced and crowdfunded translation of Herman Melville's famous novel *Moby Dick*.[3] In 2014 Microsoft added emoji support to Bing search.[4] In 2013 Katy Perry released a lyric video for her hit song *Roar*, showing the lyrics as a mixture of words and emojis.[5] We end the series of examples by a statistics from a few years ago: by 2015, already half of the Instagram posts contained at least one emoji.[6] Fig. 1 shows the opening emoji sequence (U+260E, U+1F468, U+26F5, U+1F40B, U+1F44C) of *Emoji Dick* using different sets of emojis.

[2] https://www.unicode.org/emoji/charts-14.0/emoji-counts.html

[3] http://emojidick.com/

[4] https://blogs.bing.com/search/2014/10/27/do-you-speak-emoji-bing-does/

[5] http://www.mtv.com/news/1712176/katy-perry-roar-lyric-video/

[6] https://instagram-engineering.com/emojineering-part-1-machine-learning-for-emoji-trendsmachine-learning-for-emoji-trends-7f5f9cb979ad

## 3. Emojis in natural language processing applications

With the growing popularity of emojis, these are used worldwide in numerous apps and platforms alongside text in different languages. Emojis play different roles, they offer "both complementary and supplementary relations to words" [15]. Because of this reason, many research communities took interest in emojis and their roles in written language. Although most research can be found in computer and communication science, they represent a popular research topic in marketing, behavioral science, psychology, linguistics, education, etc. as well [2].

In [7] the authors created the emoji2vec model, which – similarly to word2vec [17] – assigns continuous vector representations to emojis, based on their description: the generated emoji embeddings are the sum of the word2vec embeddings of the words from the description. The obtained emoji embeddings outperform word embeddings on the task of sentiment analysis, using a large collection of tweets.

Since in most cases emoticons and emojis are used to express emotions, attitudes, it is not surprising that they are applied most frequently in sentiment analysis, opinion mining. One of the conclusions of the analysis carried out in [26] is that a small set of emoticons have strong and clear polarity, but the rest of it, a much larger set maintain more complicated sentiments. The authors of [8] apply distant supervision to perform sentiment analysis, i.e. they use the emoticons as noisy labels, achieving high classification accuracies in the experiments. In [11] the construction of the first emoji sentiment lexicon, called Emoji Sentiment Ranking, is presented. The lexicon – which is similar to SentiWordNet [1] – contains the 751 most frequently used emojis in Twitter messages, the scores being relative frequencies in tweets having different polarities. The work [29] discusses the importance of emoticons in sentiment analysis, summarizes the existing methods, and it also briefly addresses issues such as sarcasm detection. In [10] sarcasm detection is performed by comparing the polarity of the text and of the emojis.

A video search system is presented in [4], in which video retrieval is accomplished using emojis to formulate the query. Emojis are assigned to a video based on the title, as well as by object recognition on the video frames. The closest research to ours is the Image2Emoji model presented in [3], which assigns emoji sequences to real-world images. Similarly to [4], the word2vec embeddings of the recognized objects in the pictures, as well as embeddings

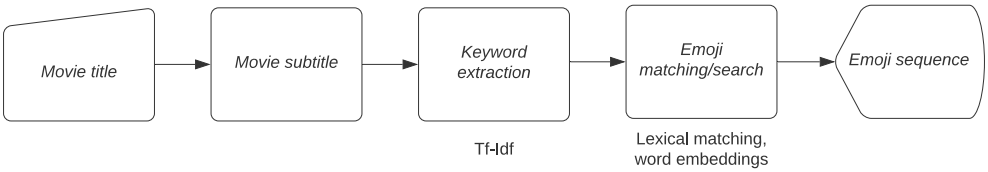



FIGURE 2. Scheme of the emoji sequence generation algorithm: given a movie title the movie subtitle is retrieved, from which the most relevant keywords are extracted, and finally, the keywords are matched with emojis using lexical matching and word embeddings to build the proper sequence.

of the picture title, description and tags are compared against the emoji embeddings to get a ranking of the most relevant emojis for a given picture.

The system proposed in this paper assigns emoji sequences to movies, using the movies' subtitles, first by extracting the keywords from the subtitle file, and then searching for the semantically nearest emojis for these.

## 4. Generating emojis from movie scripts

Emojis can be a powerful tool for natural processing algorithms to utilize. For example, the presence of a certain emoji can clearly indicate certain emotions in sentiment analysis, making such a prediction much more accurate. Sentiment analysis, however, is not the goal of this paper. We instead intend to summarize larger texts using emojis. For this purpose, we chose movie subtitles as our data source, and tried to produce a sequence of emojis to describe their plots. While at first this may seem a simple game intended only to entertain the user, summarizing a movie via summarizing its script by a handful of emojis involves many challenges. One such challenge is caused by the very small number of emojis: the retention ratio usually grows with the compression ratio [9], and in this scenario we are dealing with very low compression ratios, values around 0.0014.[7] Thus, although the goal is *just* to guess the title of the movie, it is difficult to achieve high retention ratios at such low compression rates, however, longer emoji sequences will similarly confuse the respondents.

[7]Using a random sample of 50 movie subtitles, we obtained an average number of candidate keywords of 4169.54, where a candidate keyword means that all of its characters are alphabetic, not a stopword, and having a minimum length of 3 letters. If we consider an average emoji sequence length of 6, we obtain a compression ratio of 0.0014.

In the following we will discuss the three main methods used to achieve the above-mentioned goal, our conclusions about the results and also possible future directions to expand this project. All three methods presented are based on keyword extraction and matching these keywords with existing emojis.

The scheme of the algorithm is shown in Fig. 2: we start with a movie title, obtain the subtitle of the given movie, extract its most relevant keywords, then match these keywords with emojis using different methods, and finally create the emoji sequence representing the movie.

4.1. **First approach: lexical matching.** The basis of all methods proposed is keyword extraction. For this we used *tf-idf* (term frequency $\times$ inverse document frequency) scores, known as a weighting scheme for the bag-of-words representation of documents in text categorization [23, 22], but also as a successful keyword extraction approach as well [16, 27]. This method assigns a value to every word, based on how many times it appears in the given document, and how many times it appears in other documents. A frequently used version of *tf-idf* is the following,

$$tfidf(t,d) = f_{t,d} \cdot \log\left(\frac{N}{n_t}\right),$$

where $f_{t,d}$ is the frequency of word $t$ in document $d$, $N$ is the total number of documents in the corpus, and $n_t$ denotes the number of documents containing word $t$. The more the word appears in the current document, and the fewer times in other documents, the higher the *tf-idf* value.

By this procedure, each word from a movie script will have a *tf-idf* assigned to it: the words with the highest such values are considered the keywords for a given movie. The keyword extraction step is common to all three methods presented below.

In the first method we check for emojis with names that match fully or partially with the extracted keywords. To measure the similarity between a keyword and an emoji, the Sørensen—Dice coefficient [5] was used,

$$DSC(X,Y) = \frac{2|X \cap Y|}{|X| + |Y|},$$

where $X$ and $Y$ denote two sets. More precisely, we used the convex combination of the Dice coefficients between the emoji name, treated as a set of tokens, and the extracted keywords, as well as the Dice coefficient of the keywords officially assigned to the emoji and the extracted keyword. For each extracted keyword the emoji with the highest Dice value is chosen, provided

that this value exceeds a given threshold. At the end, duplicate emojis are removed from the sequence, keeping the first occurrence only.

This approach, however, has one major drawback: if none or just a very few keywords match the emojis, the generated sequence will not be a satisfactory one. A possible solution to this would be to extract a larger set of keywords from the documents, but then the relevancy of these could be questioned.

4.2. **Second approach: word embeddings.** This approach uses vector representations to define the similarities between words and emojis. The vector representation of a word is a series of real values, obtained by applying a computational framework providing continuous word representations, e.g. *word2vec* [17], *GloVe* [20] or *fastText* [18]. Using pretrained models we can get the vector representations of the words (keywords). To do the same for the emojis, the emojis' names are used by taking the average of the vectors of the words present in the emoji name.

Similarity is computed by simply calculating the cosine of the angle enclosed by the resulting vectors [22],

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|},$$

where $\mathbf{x}^T$ denotes the transpose of $\mathbf{x}$ and $\| \cdot \|$ is the Euclidean ($\ell_2$) norm. Similarly to the previous approach, an emoji is selected if its cosine similarity is above a certain threshold, and duplicates are removed.

4.3. **Third approach: the combination of the previous two.** This third method simply combines the outcomes of the previous two methods by concatenating the resulting sequences one after the other, the first being the sequence obtained via lexical matching. Duplicate emojis are again removed from the output.

**Potential improvements.** During keyword extraction, the chronological order of the keywords is also taken into account, since a better ordering might result in a more logical enumeration. Using this method, the keywords (and later the emojis) will appear in the sequence in the same order as they appear in the movie (see Fig. 3(b)).

Furthermore, the movie title may hold a lot of information, therefore we considered *emojizing* the title too, i.e. placing the words of the movie title into the keyword list as well, applying the procedure described in Section 4.3

to the movie title (see Fig. 4). The resulting emojis – if any – are placed at the beginning of the final sequence.

## 5. Experimental results

As discussed in Section 4, we extracted the keywords from a given movie subtitle using *tf-idf* scores. Given a movie title, its subtitle is downloaded via the OpenSubtitles API[8]. The content of the subtitle file was preprocessed: tokenized, lowercased and the stopwords were removed. Next, the term frequency of each word in this document was calculated. For calculating the idf scores we used the OpenSubtitles dataset [14] (2018 version)[9]. The full English dataset contains a total of 446 612 subtitle files, but a single movie can have multiple subtitles corresponding to different versions (e.g. different authors/transcribers/translators). We used one subtitle per movie, resulting in 140 045 files processed, where the first file for each movie was taken. The lowercased dictionary of idf scores contains 1 716 310 different tokens.

To define the vector representation of the extracted keywords and the emojis the GloVe[10] model was used. It contains 400K tokens, extracted from the Wikipedia 2014 dump[11] and English Gigaword Fifth Edition[12], and their 100-dimensional vector representations. Since our method extracts no collocations but only single-word expressions as keywords, it was straightforward to calculate their GloVe representations – provided these were known tokens. In order to get continuous vector representations for the emojis, their name was tokenized, lowercased and stopword filtered, and the average of the tokens' GloVe vectors were considered. The emoji names, Unicodes and emoji keywords were obtained from EmojiNet [28]. When searching for the best matching emojis the Unicode list v13.1 was considered, omitting emojis with skin-tones,[13] resulting in a set of 1816 pictographs.

The proposed methods have a relatively large set of hyperparameters that should be selected following a systematic procedure (e.g. cross-validation), but since we did not have a usable benchmark dataset for this, we selected the following parameters in a trial-and-error fashion. Lexical matching is based on

[8] `http://www.opensubtitles.org/`

[9] `https://opus.nlpl.eu/OpenSubtitles-v2018.php`

[10] `https://nlp.stanford.edu/projects/glove/`

[11] `https://archive.org/details/enwiki-20141106`

[12] `https://catalog.ldc.upenn.edu/LDC2011T07`

[13] `http://unicode.org/emoji/charts/emoji-list.html`

TABLE 1. Table showing the extracted keywords, the most similar emojis' Unicodes and the obtained scores with the combined model + keyword ordering.

| *Interstellar* | | | *Pirates of the Caribbean* | | |
|---|---|---|---|---|---|
| **Keyword** | **Unicode** | **Score** | **Keyword** | **Unicode** | **Score** |
| ghost | U+1F47B | 0.8 | singapore | U+1F1F8 U+1F1EC | 0.8 |
| corn | U+1F33D | 0.8385 | ship | U+1F6A2 | 0.8 |
| twelve | U+1F51E | 0.8228 | fire | U+1F525 | 0.8 |
| black | U+26AB | 0.8971 | sunset | U+1F307 | 0.8 |
| hole | U+1F573 | 1.0 | mate | U+1F9C9 | 1.0 |
| fuel | U+26FD | 0.918 | fish | U+1F41F | 0.8 |
| | | | sri | U+1F1F1 U+1F1F0 | 0.9125 |

the Dice coefficient of an extracted keyword and the name and keywords of an emoji: 80% percent of the resulting score is given by the Dice similarity of the extracted keyword and the name of the emoji, and 20% of it is calculated as the Dice similarity between the extracted keyword and the keywords assigned to the emoji. The name and the emoji keywords are treated as sets, while the extracted keyword constitutes a single element set. To select a keyword/emoji into the generated sequence, a similarity threshold needs to be exceeded; we set this to 0.8 in our experiments. When working with word embeddings, the similarity threshold for cosine was also set to 0.8. The length of the generated emoji sequence is also important: being too short, there is no room for showing all the pictographs that would be good indicators for guessing the movie, while being too long could confuse the user. Both for the lexical matching and the word embedding-based methods we set this limit parameter to 6. We also used a similar limit parameter when considering the titles, setting it to 2 in both cases.

The emoji sequences obtained for the selected movies are shown in Fig. 3 and 4. Based on the test performed on a randomly chosen set of movie subtitles, the best results were obtained using the combined method, therefore we show some of the generated emoji sequences only by this approach. In order to present not only the bright side of the proposed method, we selected 3 movies for which quite decent sequences are generated, and also 3 other movies for which the extracted keywords and the generated emojis do not really make

(a)

(b)

FIGURE 3. Emoji sequences generated for 6 selected movies: (a) using the combined method, i.e. lexical maching combined with word embeddings, and (b) the same sequences shown in chronological order of the extracted keywords.

the movie guessable. The selected movies are the following: (1) *Django Unchained*[14], (2) *Interstellar*[15], (3) *Shrek*[16], (4) *Bird Box*[17], (5) *Borat*[18], (6) *Pirates of the Caribbean: at World's End*[19]. The first three of the sequences we

[14] `https://www.imdb.com/title/tt1853728`

(4) 

(6) 


FIGURE 4. Emoji sequences obtained with the combined method applying chronological ordering and placing the emojis generated from the title at the beginning of the list. Only those movies are shown here, for which considering the title as well introduced new pictographs.

consider relevant and guessable, however, the other three are less successful. Fig. 3(a) shows the emoji sequences obtained using the combined method, i.e. lexical matching combined with the word embedding-based method, Fig. 3(b) displays the same sequences but in chronological order, while Fig. 4 shows the two motion pictures for which the introduction of the title affected the results. Table 1 lists the most relevant keywords found by the combined method, together with the Unicode code points of the best fitting emojis, as well as the similarity scores obtained for two movies.

Emoji sequence generation for movie scripts was implemented in Python, and its source code can be found at `https://github.com/bajcsianna/movie2emoji`.

5.1. **Evaluating the emoji sequences.** Evaluating the generated emoji sequences, that is evaluating the performance of the proposed method proved to be a difficult task. Since generating pictograms that show the main events and motifs of a movie can be considered a summarization task, using the ROUGE metric might seem appropriate [13]. The problem with the application of this measure is twofold: (i) no sufficiently large dataset of human-generated movie emoji sequences is available, (ii) the sequences – usually containing 2 to 6 pictograms – are too short for this metric.

In order to obtain an evaluation of the proposed methods, we randomly generated 10 + 10 pictograph sequences using the combined method (without

[15] `https://www.imdb.com/title/tt0816692`
[16] `https://www.imdb.com/title/tt0126029`
[17] `https://www.imdb.com/title/tt2737304`
[18] `https://www.imdb.com/title/tt0443453`
[19] `https://www.imdb.com/title/tt0449088`

TABLE 2. Accuracies obtained for the two movie emoji quizzes, the last line showing the overall accuracy.

| | **1st quiz** | **2nd quiz** |
|---|---|---|
| **Movie #1** | 13.15% | 31.88% |
| **Movie #2** | 0% | 23.18% |
| **Movie #3** | 0% | 57.97% |
| **Movie #4** | 0% | 30.43% |
| **Movie #5** | 0% | 50.72% |
| **Movie #6** | 36.82% | 26.08% |
| **Movie #7** | 2.63% | 68.11% |
| **Movie #8** | 65.78% | 30.43% |
| **Movie #9** | 26.31% | 46.37% |
| **Movie #10** | 44.73% | 27.53% |
| | 18.94% | 39.27% |

(a)

(b)

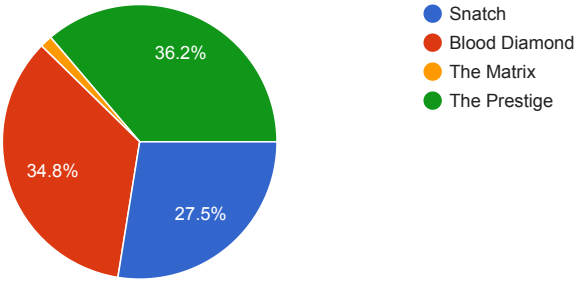



FIGURE 5. Example for an overly successful obfuscation: (a) emoji sequence obtained for *Snatch* and (b) the answers given by the participants. The results show that three of the four possible answers seemed equally acceptable for the participants of the survey.

taking into account the movie title), and assembled two surveys: ($s_1$) for guessing the movies by typing in the title, and ($s_2$) for guessing the movies in form of multiple choice questions with four possible answers. The selected movies contain only motion pictures and animated films, not necessarily blockbusters,

released between 1994 and 2019. The surveys have been sent to students of the Babeş–Bolyai University and members of the academic staff via learning management systems, business communication platforms and social networking sites. For the two quizzes we received 38 and 69 answers, respectively.

The accuracy results are shown in Table 2. As it was anticipated by us, the second quiz produced better results, since the possible answers were narrowed down to four, as in the case of the first quiz all existing films – from the abovementioned period, as this was brought to the attention of the respondents – had to be considered. For the first quiz the majority answers correctly determined 3 out of 10 movies, while this number was 4 for the second quiz.

In survey ($s_2$), in addition to the correct title we selected three incorrect ones, but not in a random manner: the incorrect titles were chosen such that at least one emoji matched the main motifs of the movie. However, this obfuscation sometimes worked too well. In Fig. 5(a) the emojis generated for the movie *Snatch*[20] are shown, while the pie chart in (b) shows the distribution of the participants' answers. The answers for other movies show the signs of too successful obfuscation as well, which certainly affects the results. Choosing the movies to generate the emoji sequences for is also a difficult task, since one cannot assure that all movies are known by the participants. Similarly, one cannot expect the same degree of seriousness of quiz completion from all participants. Therefore, we suggest to consider the obtained accuracy scores as lower limits of guessability of the generated emoji sequences.

The quizzes used in the evaluation process – with the correct answers – are available at the following links: movie2emoji I.: `https://forms.gle/YHGeky6oCcxAwjkd8`, movie2emoji II.: `https://forms.gle/XFkjCHtZXbcbkjbN8`.

## 6. Discussion and future work

In this paper we presented a system that is able to assign emoji sequences to movies, based on the movie's subtitle. The pipeline of the proposed method is simplistic but rather effective: extraction of the most relevant keywords from the subtitle (or script) of the movie, and then assigning emojis to these. We experimented with three approaches: (i) lexical matching using Dice coefficients to determine similarity, (ii) a word embedding-based approach using cosine similarity, and (iii) the combination of these two.

[20] `https://www.imdb.com/title/tt0208092`

While the obtained results are promising, there is room for improvements and further experiments too. Since the keyword extraction model applied is a central component of this approach, we consider experimenting with other such models important. In the present system information is acquired only from the movie subtitle, providing an efficient means to generate the emojis, which could be supplemented by object recognition models considering also the movie frames when available, similarly to [3]. Studying the effect of stemming/lemmatization of the extracted keywords on the output is also left as a future work. Part of speech tagging and selection of words belonging to important parts (nouns, verbs, adjectives), as well as considering word collocations or neighborhoods of the selected keywords can also positively affect the performance of our system.

## Acknowledgements

We would like to thank the students and the academic staff of the Babeş–Bolyai University who participated in our experiments to evaluate the proposed models.

## References

[1] Baccianella, S., Esuli, A., and Sebastiani, F. SentiWordNet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *LREC* (2010), vol. 10, pp. 2200–2204.

[2] Bai, Q., Dan, Q., Mu, Z., and Yang, M. A systematic review of emoji: Current research and future perspectives. *Frontiers in Psychology 10* (2019), 2221.

[3] Cappallo, S., Mensink, T., and Snoek, C. G. Image2emoji: Zero-shot emoji prediction for visual media. In *Proceedings of the 23rd ACM International Conference on Multimedia* (2015), pp. 1311–1314.

[4] Cappallo, S., Mensink, T., and Snoek, C. G. Query-by-emoji video search. In *Proceedings of the 23rd ACM International Conference on Multimedia* (2015), pp. 735–736.

[5] Dice, L. R. Measures of the amount of ecologic association between species. *Ecology 26*, 3 (1945), 297–302.

[6] Dresner, E., and Herring, S. C. Functions of the nonverbal in CMC: Emoticons and illocutionary force. *Communication Theory 20*, 3 (2010), 249–268.

[7] Eisner, B., Rocktäschel, T., Augenstein, I., Bošnjak, M., and Riedel, S. emoji2vec: Learning emoji representations from their description, 2016.

[8] Go, A., Bhayani, R., and Huang, L. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford 1*, 12 (2009), 2009.

[9] Hovy, E. Text summarization. In *The Oxford Handbook of Computational Linguistics*, R. Mitkov, Ed. Oxford University Press, Oxford, 2004, ch. 32.

[10] Karthik, V., Nair, D., and Anuradha, J. Opinion mining on emojis using deep learning techniques. *Procedia Computer Science 132* (2018), 167–173.

[11] Kralj Novak, P., Smailović, J., Sluban, B., and Mozetič, I. Sentiment of emojis. *PloS One 10*, 12 (2015), e0144296.

[12] Kumari, R., and Gangwar, R. Use of expression based digital pictograms in interpersonal communication: a study on social media and social apps. *International Journal of Innovative Knowledge Concepts 6* (2018), 11.

[13] Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out* (2004), ACL, pp. 74–81.

[14] Lison, P., and Tiedemann, J. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)* (Portorož, Slovenia, May 2016), European Language Resources Association (ELRA), pp. 923–929.

[15] Mei, Q. Decoding the new world language: Analyzing the popularity, roles, and utility of emojis. In *Companion Proceedings of The 2019 World Wide Web Conference* (New York, NY, USA, 2019), WWW '19, Association for Computing Machinery, p. 417–418.

[16] Mihalcea, R., and Csomai, A. Wikify! Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management* (2007), pp. 233–242.

[17] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space, 2013.

[18] Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., and Joulin, A. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)* (2018).

[19] Nenkova, A., and McKeown, K. A survey of text summarization techniques. In *Mining Text Data*. Springer, 2012, pp. 43–76.

[20] Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014), pp. 1532–1543.

[21] Radford, W., Chisholm, A., Hachey, B., and Han, B. :telephone::person::sailboat::whale::okhand:; or "Call me Ishmael" – How do you translate emoji? In *Proceedings of Australasian Language Technology Association Workshop* (2016), pp. 150–154.

[22] Schütze, H., Manning, C. D., and Raghavan, P. *Introduction to information retrieval*. Cambridge University Press, 2008.

[23] Sebastiani, F. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR) 34*, 1 (2002), 1–47.

[24] Stark, L., and Crawford, K. The conservatism of emoji: Work, affect, and communication. *Social Media + Society 1*, 2 (2015), 2056305115604853.

[25] Taggart, C. *New words for old: Recycling our language for the modern world*. Michael O'Mara Books, 2015.

[26] Wang, H., and Castanon, J. A. Sentiment expression via emoticons on social media. In *International Conference on Big Data* (2015), IEEE, pp. 2404–2408.

[27] Wartena, C., Brussee, R., and Slakhorst, W. Keyword extraction using word co-occurrence. In *International Workshops on Database and Expert Systems Applications* (2010), IEEE, pp. 54–58.

[28] Wijeratne, S., Balasuriya, L., Sheth, A., and Doran, D. EmojiNet: An open service and API for emoji sense discovery. In *Proceedings of the International AAAI Conference on Web and Social Media* (2017), vol. 11.

[29] Yadav, P., and Pandya, D. Sentireview: Sentiment analysis based on text and emoticons. In *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)* (2017), pp. 467–472.

Faculty of Mathematics and Computer Science, Babeş–Bolyai University, Cluj-Napoca, Romania

*Email address*: anna.bajcsi@stud.ubbcluj.ro

*Email address*: barbara.botos@stud.ubbcluj.ro

*Email address*: peter.bajko@stud.ubbcluj.ro

*Email address*: zbodo@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXVII**, Number 1, 2022
DOI: 10.24193/subbi.2022.1.02

# MULTIPLE TYPES OF AI AND THEIR PERFORMANCE IN VIDEO GAMES

IULIAN PRĂJESCU AND ALINA DELIA CĂLIN

Abstract. In this article, we present a comparative study of Artificial Intelligence training methods, in the context of a racing video game. The algorithms Proximal Policy Policy Optimization (PPO), Generative Adversarial Imitation Learning (GAIL) and Behavioral Cloning (BC), present in the Machine Learning Agents (ML-Agents) toolkit have been used in several scenarios. We measured their learning capability and performance in terms of speed, correct level traversal, number of training steps required and we explored ways to improve their performance. These algorithms prove to be suitable for racing games and the toolkit is highly accessible within the ML-Agents toolkit.

## 1. Introduction

From their inception in the 1950s, video game started to evolve and to become more and more complex in terms of better graphics, interaction controllers and game mechanics, audio and visual feedback, progressing at the same pace with the technology of the time and sometimes even pushing technology forward, becoming the beautiful pieces of art we think about today. The industry has been dominated by a small number of companies that established specific practices around the development and distribution of video games. Strategy video games have found an important role due to their effect on improving hand–eye coordination and visual-motor skills [14].

One of the most important steps in this evolution is marked by adding Artificial Intelligence (AI) methods, which simulates the presence of other players or characters, increasing the immersive experience of the game. This aims at designing agents capable of playing video games without human intervention [12], often called non-player characters. Thus, the efficiency of an AI agent in a game is generally evaluated by human experience [9]. The importance of

Received by the editors: 23 September 2021.

2010 *Mathematics Subject Classification.* 91A10, 68T05.

1998 *CR Categories and Descriptors.* I.2.1 [**Artificial intelligence**]: Applications and Expert Systems – *Games*; K.8.0 [**Personal computing**]: General – *Gaming*.

*Key words and phrases.* racing game, PPO, GAIL, behavioral cloning, AI in games.

AI tools in games is not limited to the game experience, but provides a rich research ground for studying and experimenting how humans interact with AI agents [21].

However, there is a gap between academic and industrial approach of game AI that needs addressing. The basic AI algorithms usually used in games (such as ad hoc authoring, tree search, evolutionary computation, and machine learning) do not rise to the current demands, meaning that new methods and techniques are needed [4].

In this paper we aim to encourage a more sophisticated use of AI in industry (such as neural networks) [20], by presenting the new tools available (like the Unity ML-Agents Toolkit, Pytorch) and specific case scenarios where algorithms can be used successfully. Thus, this paper focuses on determining the best method to train AI agents for a specific type of game: car racing simulation games, by presenting a specific video game context. The importance of this type of game is not only recreational, but given the realistic environment, it is used to develop driving skills in a safe environment. Using intelligent methods to simulate the required challenges of the environment and drive progress by competition with non-human agents is the key to success. Three types of AI agents are compared in simulating car driving agents, using the game development platform Unity [8] and the machine-learning agents module based on the PyTorch technology [11]: Proximal Policy Optimization (PPO), Behavioral cloning (BC) and Generative Adversarial Imitation Learning (GAIL) algorithms [18].

## 2. Background

Some examples of related work in the field would include the idea of a unified video game AI middleware [15], which was created by The International Game Developers Association (IGDA) by launching an Artificial Intelligence Interface Standards Committee (AIISC) in 2002, which had the goal of creating a standard AI interface for reusing and outsourcing AI code [15]. In Berndt et al. [1], was proposed an Open AI Standard Interface Specification (OASIS), which aimed at making the integration of AI in video games easier. This kind of game AI middleware can now be found in multiple video game engines [15], such as CryEngine, Havok, Unreal Engine and Unity, these game engines aiming to provide realistic agents and virtual environments.

In relation to racing games, recent interest has been present in the literature with the most focus on algorithms such as PPO for vehicles in mixed and full-autonomy traffic [13, 17], GAIL for modelling a human driver [2, 10], or BC for robust autonomous vehicles with end-to-end imitation learning [16].

2.1. **The Unity ML-Agents Toolkit.** This Unity toolkit is an open source project that consists of two elements: the ML-Agents software development kit (for creating environments within the Unity Editor and with the associated C# scripts) and a Python package (to help interfacing with the environments created). ML-Agents presents three components: (1) the Agent, responsible with collecting observations and taking actions; (2) the Brain, responsible with making decisions for the linked Agents containing matching observation and action space configurations; (3) the Academy, responsible for managing the learning environment by keeping track of the steps performed by the Agents, setting the target simulation speed and frame rate, and resetting parameters for eventual configuration changes during run-time.

The Python Unity ML-Agents Trainers Package provided in this toolkit communicates with Unity by using the included UnityEnvironment class, by the use of a gRPC communication protocol, which utilises protobuf messages.

2.1.1. *Proximal Policy Optimization (PPO).* By trying to improve the already ample scene of reinforcement learning with neural network function approximators, the OpenAI team introduces a new family of policy gradient methods with the Proximal Policy Optimization Algorithms [18]. These new methods share some of the benefits brought by the trust region policy optimization (TRPO), but have the advantage of having better sample complexity (empirically). While TRPO uses a complex second-order method when confronted with the problem of trying to improve the step on a policy using the data it currently has without stepping too far as to cause a performance collapse, PPO uses a family of first-order methods which use some other algorithmic approaches to keep the new policies close to old.

The main deviations of PPO are the PPO-Penalty and the PPO-Clip. We will primarily focus on the PPO-Clip variant as it is the most commonly used and it is present in the ML-Agents toolkit used in this study. As opposed to PPO-Penalty, it does not have any constraint or a KL-divergence term in the objective, but instead relies on specialised clipping in the objective function to remove incentives for the new policy to get far from the old policy. The PPO algorithm uses fixed-length trajectory segments, where on each iteration, every of the N actors collect T timesteps of data in parallel, then constructs the surrogate loss on the NT timesteps of data and optimises them with minibatch Stochastic gradient descent (SGD) for a $K$ number of epochs.

2.1.2. *Generative Adversarial Imitation Learning (GAIL).* Generative Adversarial Imitation Learning (GAIL) is an Inversive Reinforcement Learning algorithm, which, as the name suggests, uses a Generative Adversarial Network (GAN) to function. This algorithm can be also described as a model-free

imitation learning algorithm, and can yield a good performance for complex behaviours, particularly in big, high-dimensional environments. As presented in [6], GAN is a type of generative model, which brings a way to learn deep, hierarchical representations in a semi-supervised or unsupervised manner.

The GAN architecture consist of two different networks working together to learn from existing datasets. The first network is the generator, which has the role of generating new data by learning the distribution of the input dataset. The second network called the discriminator has the role of gathering the samples from the training data and classifying them either as generated by the generator or as real data. The Inversive reinforcement learning (IRL) methods were presented in the idea of helping the reinforcement learning agents to learn the experts policy and to get reward functions in order to explain the experts behaviors from their given trajectory [7].

2.1.3. *Behavioral Cloning (BC).* Behavioral Cloning (BC) represents a form of "Imitation Learning" which has the goal of creating a model of a human's behavior when trying to execute a difficult set of actions. The BC method is one of the most used approaches in regards to the imitation learning problem and has been proven powerful in the sense that it can very quickly imitate the demonstrator without needing to interact with the environment [19].

This method has been used in many different applications, from flying down a quadrotor on a forest trail [5], to autonomous driving [3]. BC is related with other methods of learning by imitation [31], such as GAIL [7], IRL and other methods that use data from human performance. The behavioral cloning algorithm used by the ML-Agents toolkit is one of Behavioral Cloning from Observation [19] and works in the following fashion: the algorithm needs to find a good imitation policy from a set of state-only demonstration trajectories. The extraction of the agent-specific part of the demonstrated state sequence and the forming of a set of demonstrated agent-specific state transitions, in order for the use of the agent-specific inverse dynamic model [19]. For each transition the algorithm computes the model-predicted distribution over demonstrator actions and uses the maximum-likelihood action as the inferred action. We then build the set of complete state-action pairs [19].

## 3. Case study

3.1. **The game.** The racing game environment we study is built in Unity, employing several levels (tracks), race configurations, and car models (see Figure 1). The player can compete against multiple AI cars trained and compared in this case study. The agents used different training methods such as PPO, GAIL, BC, and Soft Actor-Critic (SAC). After multiple training sessions, the

SAC models did not manage to train to the point of completing the level so the other methods were used forward.


FIGURE 1. Game circuit

3.1.1. *Training the AI.* After the training process, five models using three different training methods were obtained, one type using PPO, two types using GAIL and two types using BC. The training of the AIs was done using ML-Agents. In this framework the necessary components for training an agents include a virtual environment, the agent component present in the Unity project and a configuration file which holds all the variables and parameters of the neural network and training method used.

The specific settings for each training methods are as follows: for the PPO method the `trainer_type: ppo` and the `reward_signals` need to be `extrinsic`; for the GAIL methods one more module needs to be added, which is the `gail:` one with multiple specific parameters such as `demo_path` for showing the location of the demo file used in the training process, `strength` value representing how much the agent should copy the demonstration, `gamma`, `learning_rate`, `use_actions` and `use_vail`; finally for the BC method the `behavioral_cloning` module which adds the `demo_path` parameter and the `value` representing how much the agents should copy the demonstration and other BC specific parameters.

## 3.2. **Experiments.**

3.2.1. *Training PPO.* The parameters of the configuration file were adjusted to train multiple agents using trial and error in order to increase the performance of the agents. The best configuration identified (Test 33), which consisted of 40 agents, used a `batch_size` of 120, the `learning_rate` of 0.0003

and `strength` of the `extrinsic` `reward_signals` of 1(the maximum recommended value). This configuration demonstrated very good performance in accurately parsing the circuit, with the maximum speed reaching the value of 27. The agents started training with the environment reward at -1.477 and after training for over 5 million steps they reached the value of 0.7687, being the highest value achieved with this method, and it had a growth value of 2.2457. The evolution of the training is shown in Figure 2.

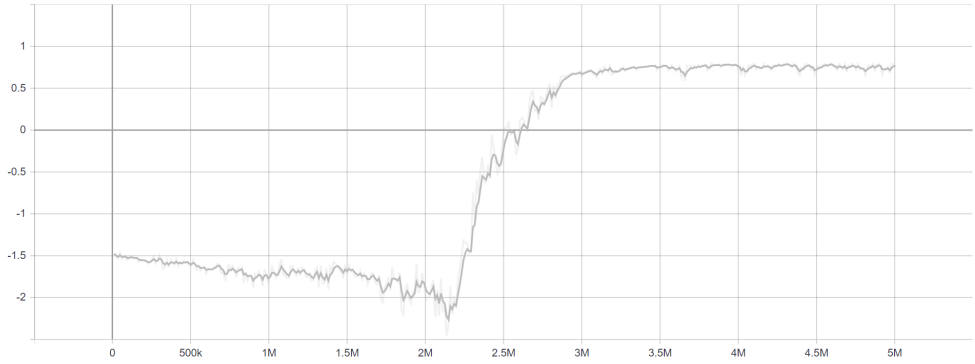



FIGURE 2. Environment cumulative reward of the PPO method

In terms of traversing the level, the final model takes shortcuts by cutting corners and getting off the track portion of the level in order to complete the level as fast as possible. This is a good thing in the context of finishing first, but ultimately decreases the value of the model for not traversing the level correctly.

3.2.2. *Training GAIL.* For the GAIL training method two types of demos were used, one made by a human player, and one made using the PPO trained method, where one demonstration of traversing the track and one agent using the PPO Test 33 brain were recorded. This was done in order to determine if there is a difference in performance between these two kinds of demos.

For both models trained using the GAIL method, along with the 40 agents used, the extrinsic `reward_signals` module was utilised, with the same values as the PPO method, in collaboration with the `gail` module, which included the `learning_rate` of 0.0003, the `encoding_size` of 128 and the gail `strength` of 0.1. After adding the `gail` module, the agents started to learn and the cumulative reward started increasing alongside the performance on the racing track.

The performance of the GAIL method using a player demo reached the speed value of 21, being a very good one for the gaming context, achieved after a bit over 5.5 million steps, and after starting with the environmental cumulative reward of -1.486, it reached the value of 0.725, having a growth value of 2.211. The evolution of this method can be seen in Figure 3.

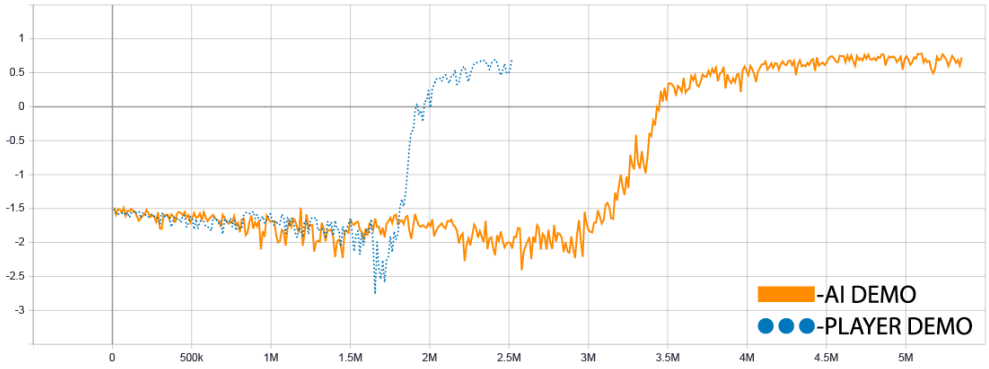



FIGURE 3. Environment cumulative reward of the GAIL method using a player demo versus using an AI demo

The performance of the GAIL method using the AI demo reached the speed value of 17 after just over 2.5 million steps. Starting with the cumulative reward of -1.486 and reaching the value of 0.679, it had an approximated growth value of 2.165, very close to the player made demo method described above and also the PPO model, used for creating the demo after which this model learned. We will examine the evolution of the training session present in Figure 3.

When comparing the two GAIL methods we can see that they have similar results but also big differences. Starting with the similarities, they both have very close growth values and the learning process is very similar, both with the cumulative reward slowly decreasing until the second half of the session where they started to reach their maximum value very fast, then very slightly increasing until the end of the training session. Considering this, the second method, using the AI made demo, learned twice as fast as the first one, but ultimately had a smaller speed performance.

With all this said, we can see that using the player made demo was better than the AI made one, even after considering the inefficiency in time.

3.2.3. *Training BC.* Just like the GAIL agents, the BC method was used to create two types of AI using the same demos as before to determine if the type of demo affects the performance of the AIs and what differences can be

found. The best configuration found so far for this method used 40 agents and the same values for the `extrinsic` `reward_signal` as the PPO and GAIL methods, with the exception of the `batch_size` which was increased to 512 and the behavioral cloning module was added which included a `strength` of 0.1. The best session of the BC method using a player demo was the fourth one with the maximum speed of 6. This result was achieved after 6 million steps, starting with the environment cumulative reward of -5.534 and reaching the value of -4.053 by the end of the training session. The growth value of this method was 1.481 and we can see the evolution of the agent in Figure 4.

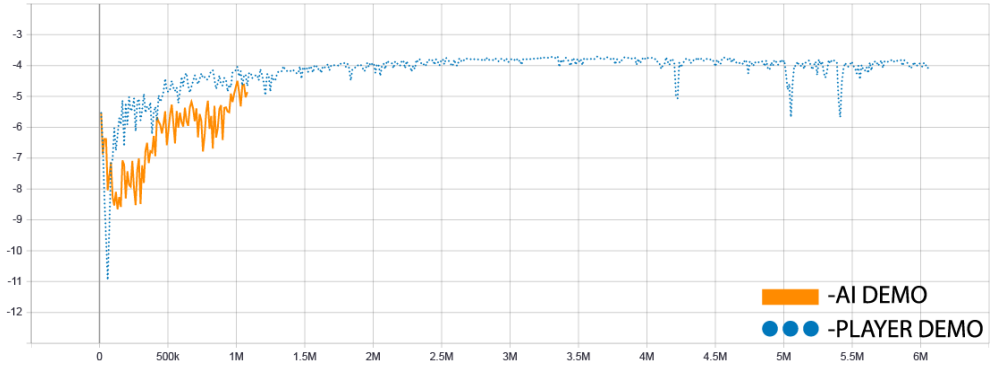



FIGURE 4. Environment cumulative reward of the BC method using a player demo versus using an AI demo

Unfortunately, when traversing the environment, this model also takes shortcuts, going off the track part and onto the surrounding environment, losing value.

As the aforementioned GAIL agent that used an AI made demo, this one also uses the same demo made from the performance of the PPO trained AI. Just like the agents trained with a player demo, the AI demo trained agents achieved a speed performance value of 6 but in this case, the training session was much shorter, ending after just over 1 million steps. Within this period, the agents grew the environment cumulative reward from -5.531 to the value of -4.863, having the final growth value of 0.668.This method had the poorest growth while training and we can visualise it in Figure 4.

Even though both models using the BC training method had the same maximum speed value, the difference between these two methods is the one of training session length and efficiency, the agents using the AI made demo reached the same performance almost 6 times faster than the one trained with the player made demo even though the latter had a bigger growth value.

Overall, we can say that using an AI made demo is better than using a human player demo for the BC training method.

3.3. **Results.** When comparing all five models trained for this experiment, we can see exactly how different the training methods perform and which one has the best performance.

The methods with the fastest growth of their reward value are the BC methods (Player demo red, AI demo dark blue) reaching values close to maximum in just 1 million steps, after that, the GAIL method using an AI made demo (green) is in third place, followed by the PPO method (orange) and finally the GAIL method using a player made demo (light blue).

For the criteria of correctness while traversing the level, all five models have a bad performance, taking shortcuts and cutting corners through the level by going off the track onto the surrounding environment (leading to incorrect level traversal). This fact will not be taken into consideration in the current comparison.

All results are compared in Table 1 below, in terms of speed of circuit traversal, growth value (based on starting and ending reward) and number of steps involved in the training.

TABLE 1. Comparison of all five initial AIs specifications

| | **Speed** | **Starting reward** | **Ending reward** | **Growth value** | **No. of steps** |
|---|---|---|---|---|---|
| **PPO** | 27 | -1.477 | 0.7687 | 2.2457 | 5 mil |
| **GAIL& player demo** | 21 | -1.486 | 0.725 | 2.211 | 5.5 mil |
| **GAIL & AI demo** | 17 | -1.486 | 0.679 | 2.165 | 2.5 mil |
| **BC & player demo** | 6 | -5.534 | -4.053 | 1.481 | 6 mil |
| **BC & AI demo** | 6 | -5.531 | -4.863 | 0.668 | 1 mil |

This table shows us all the properties of each model in the order of which they were trained. Coincidentally, the order also represents the performance order of the models. The PPO model had the best performance of all the trained AIs with the biggest speed and growth values. The next best performance is of the GAIL method with both models having a good performance and as stated in subchapter 3.2.2, the model using a player made demo had a better performance than the one using the AI made demo in both speed and growth value.

The method with the least performance is the BC one, with both models reaching the low speed value of 6 and having suboptimal growth values compared to the other two methods. Even though the BC model using the AI agent demo has the smaller growth value than the one trained with a a human

player demo, it managed to train about 6 times faster reaching the same speed performance, therefore we conclude that it is a much better model.

### 3.4. **Improvements.**

3.4.1. *Improving PPO.* For this training session, the parameters `batch_size` and `learning_rate` were increased to 2048 and 0.0005 respectively and 20 agents were used, which at the beginning of the session had a cumulative environment reward of -1.562 and after just 5 million steps, reached the reward value of 0.7193, having an approximate growth value of 2.2813 and an average speed value of 24.

In Figure 5, we can see how the second version (blue) started the training session very close to the first one and oscillated until the 1.7 million steps mark, compared to the 2 million steps mark of the first version (grey). After that point, it slowly started to learn, oscillating until the 3.5 million steps mark where it reached its maximum potential and until the end of the episode maintained its value close to the maximum like the first version.

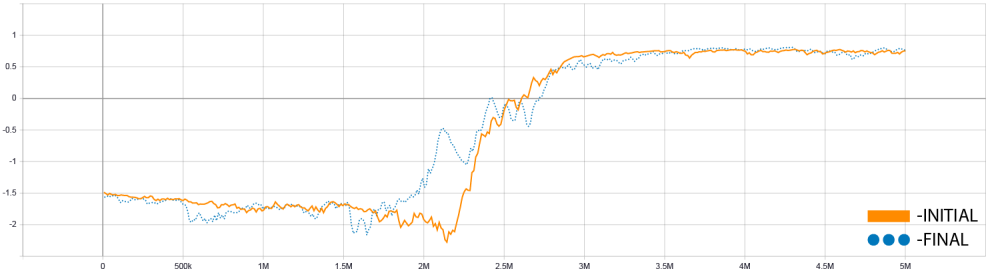



FIGURE 5. Environment cumulative reward of the PPO method initial versus final model

When comparing the first model with the improved one, the improved one has a slower speed value, 24 versus 27, but a slightly bigger growth value 2.2813 versus 2.2457. While traversing the level, the improved model has a better understanding of the environment, maintaining its traversing pattern almost exclusively on the track part of the level, compared to the first model which cuts corners in order to complete the level faster. This adds more value to the improved model, making it more realistic and better suited for this genre of video games.

3.4.2. *Improving GAIL.* Compared to the first models trained with the GAIL method, the improved ones used 20 agents, new demos and had the same configuration with only a slight increase in the gail `strength`, having the value of 0.15. The performance of the improved GAIL model using a player made

demo is very good, reaching the speed value of 21 after 5 million steps and after starting with the cumulative environment reward of -1.486, it reached the value of 0.8046, having an approximate growth value of 2.2906.

In Figure 6 we can see the training evolution of the model, compared to the previous version.

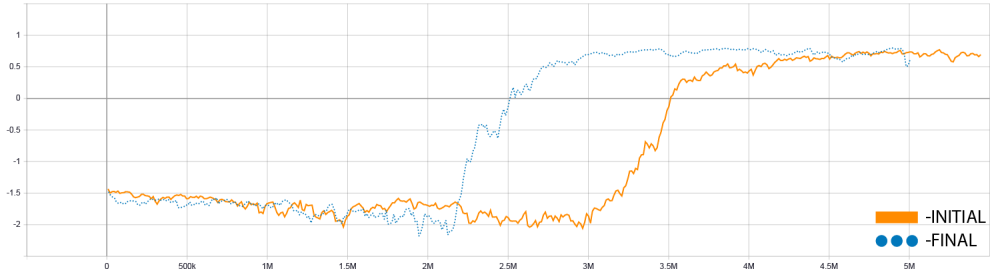



FIGURE 6. Environment cumulative reward of the GAIL method using a player demo initial versus final model

As we can see, the improved model (orange) learned faster due to the increase in the learning rate, from around the 2.1 million steps mark compared to the 3 millions steps mark of the first version (blue). It reached its maximum potential around the 3 million steps mark and from there, maintaining its value close to its maximum until the end of the session.

Both the first and the improved models have relatively the same performance, with average speed of 21, but the improved model has a slightly bigger growth value. When comparing the models while traversing the level, the improved one has a better understanding of the environment, traversing the level almost exclusively on the track part. This again adds more value to the improved model, making it more realistic and better suited for this genre of video game.

The performance of the improved GAIL model using an AI made demo is very good, reaching the speed value of 21 after 5 million steps and after starting with the cumulative environment reward of -1.527, it reached the value of 0.7133, having an approximate growth value of 2.2403.

In Figure 7 we can see the training evolution of the model, compared to the initial experiment. When comparing the two versions, the improved one has an approximately 23% increase in speed performance, going from 17 to 21, and it has a bigger growth value. While traversing the level, both the models take shortcuts, going off the track part and on to the surrounding environment, so there is no significant improvement in this department.

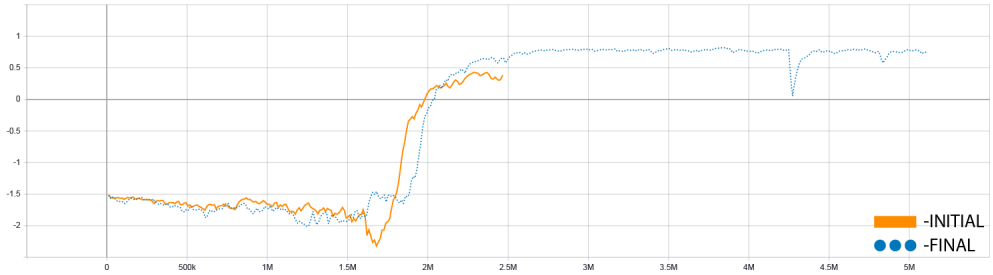



FIGURE 7. Environment cumulative reward of the GAIL method using an AI demo initial versus final model

3.4.3. *Improving BC.* Regarding the improved BC configuration, the only differences consisted of using 20 agents, just like the improved PPO and GAIL models, and using new demos from which the agents learned.

The performance of the improved BC model using a player made demo did not increase in terms of speed, having the average speed of 6, like the model before it. In terms of environmental cumulative reward, it started with the value of -1.499 and had a maximum value of 0.4614 with an approximate growth value of 1.9604.

As we can see from Figure 8, the improved version of this method started to slowly learn until reaching close to its maximum potential at the 1 million steps mark. From there until the 2.5 million steps mark held its value very steady, but after that it had an unpredictable behaviour and slowly decreased until the end of the training session.

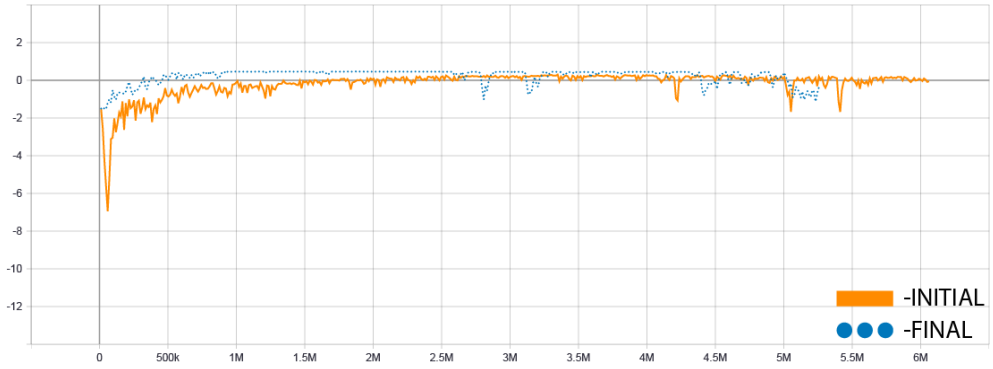



FIGURE 8. Environment cumulative reward of the BC method using a player demo initial versus final model

When compared to the previous model, the improved one has the same average speed but a bigger growth value, 1.9604 versus 1.481. While traversing the level, the improved model learned to take fewer shortcuts, by cutting less corners and staying more on the track part of the level, this in turn increases the value of the model.

The performance of the improved BC model using an AI made demo did not increase in terms of speed, having the average speed of 6, like the model before it. In terms of environmental cumulative reward, the model started with the value of -1.541 and had a maximum value of 0.5802 after 5 million steps with an approximate growth value of 2.1212.

In Figure 9 we can see how the improved model started to learn slowly, reaching its maximum potential at around the 1 million steps mark and keeping its value pretty consistent throughout the training session, until approximately around the 3.3 million steps mark where it started to raise and fall until the end of the session. Compared to the previous version it had a more stable learning rate, the first version having a very unpredictable learning pattern.

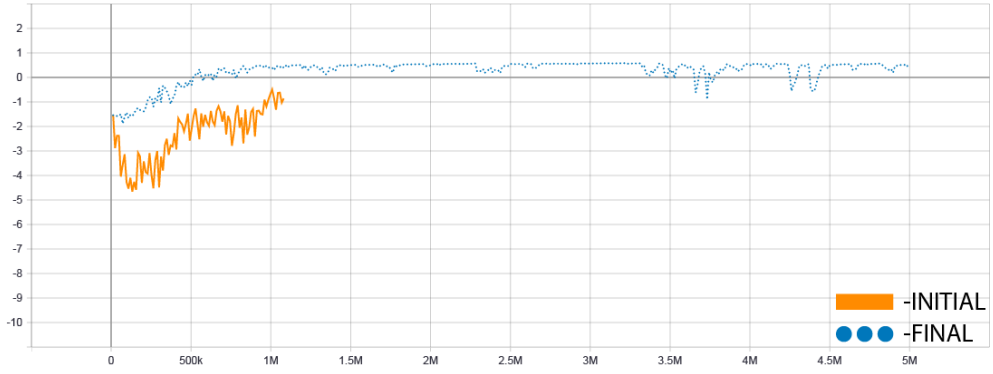



FIGURE 9. Environment cumulative reward of the BC method using an AI demo initial versus final model

When compared to the previous model, the improved one has the same average speed but a bigger growth value, 2.1212 versus 0.668. While traversing the level, the improved model follows along relatively the same pattern as the previous version, taking shortcuts, leaving the track part of the level and traversing the surrounding environment. Other than the increase in growth value, the model did not receive an increase in its value compared to the previous version.

3.5. **Final results.** When comparing the improved version of the five models trained for this experiment, we can see even more clearly how different the training methods perform and which one has the best performance.

The method with the fastest growth is the BC method (red for BC using a player demo and dark blue for BC using an AI demo), with both its models reaching their maximum potential by the 1 million steps mark. Following we have the GAIL model using an AI demo (brown), reaching its maximum at around the 2.5 million steps mark. Finally, we have the PPO (light blue) and GAIL using a player demo (orange) models, which both reached their maximum around the 3 million steps mark.

In the context of traversing the environment, the models of PPO, GAIL using a player demo and BC using a player demo have improved by staying more on the track and not taking shortcuts on their way to complete the level, with the BC model only having a slight improvement in this regard.

In Table 2, we have the values of the performance for all the five improved models (with 5 mil number of steps for each) with correct level traversal for some (does not cut through the environment like before). The best model was the one of the PPO method, which had the biggest speed value out of all five models, with the value 24. The PPO model is followed again by the GAIL models, both having the speed value of 21. Lastly, the BC models had again the poorest performance with the average speed value of 6.

Table 2. Comparison of all five improved AIs specifications

| | **Speed** | **Starting reward** | **Ending reward** | **Growth value** | **Correct level traversal** |
|---|---|---|---|---|---|
| **PPO** | 27 | -1.562 | 0.7193 | 2.2813 | Yes |
| **GAIL& player demo** | 21 | -1.486 | 0.8046 | 2.2906 | Yes |
| **GAIL & AI demo** | 21 | -1.527 | 0.7133 | 2.2403 | No |
| **BC & player demo** | 6 | -1.499 | 0.4614 | 1.9604 | Yes |
| **BC & AI demo** | 6 | -1.541 | 0.5802 | 2.1212 | No |

The reason behind this ranking is that the PPO model had the best speed performance in both the original and improved model, with the increase in value coming from the improved model, which learned to traverse the environment more correctly. Next, we have the GAIL models, which had the same speed performance in the improved model, but ultimately the model using a player made demo learned to traverse the environment more correctly than the model using an AI made demo. For the final places, the BC method had the poorest performance of all five model, but the model using a player made

demo learned a little bit better to traverse the environment and ultimately this put it at an advantage compared to the one using an AI made demo.

## 4. Conclusions

The purpose of this article is firstly to present the power of the ML-Agents toolkit, which, as we have seen, is a very competent and accessible tool for training multiple types of intelligent agents using different training methods. This is thanks to the use of a high-level framework such as PyTorch, working in the background. This is a facilitating tool in the process of creating, training and adding artificial intelligence to video games, supporting the game development industry.

Moreover, we have shown how different methods of AI perform compared to one another in the context of a racing video game and which would be the best option to choose when developing this type of video games. The PPO method, using reinforcement learning, had the best performance of all the trained models, followed by GAIL and BC respectively. The results found in this experiment may not be definitive, as there is always room for improvement and every training game environment is different, but they are a good reference point on how each of these methods performs. The implemented application also shows the simplicity and efficiency of the training process and it is a very good graphical representation of the results found in this experiment.

## References

[1] Berndt, C., Watson, I., and Guesgen, H. Oasis: an open ai standard interface specification to support reasoning, representation and learning in computer games. In *IJCAI-05 Workshop on Reasoning, Representation, and Learning in Computer Games* (2005), Citeseer, pp. 19–24.

[2] Bhattacharyya, R., Wulfe, B., Phillips, D., Kuefler, A., Morton, J., Senanayake, R., and Kochenderfer, M. Modeling human driving behavior through generative adversarial imitation learning. *arXiv preprint arXiv:2006.06412* (2020).

[3] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[4] Fan, X., Wu, J., and Tian, L. A review of artificial intelligence for games. *Artificial Intelligence in China* (2020), 298–303.

[5] Giusti, A., Guzzi, J., Cireşan, D. C., He, F.-L., Rodríguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Caro, G. D., Scaramuzza, D., and Gambardella, L. M. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters 1*, 2 (2016), 661–667.

[6] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. *Advances in neural information processing systems 27* (2014).

[7] Ho, J., and Ermon, S. Generative adversarial imitation learning. *Advances in neural information processing systems 29* (2016), 4565–4573.

[8] Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., et al. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627* (2018).

[9] Kreminski, M., Samuel, B., Melcer, E., and Wardrip-Fruin, N. Evaluating ai-based games through retellings. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (2019), vol. 15, pp. 45–51.

[10] Kuefler, A., Morton, J., Wheeler, T., and Kochenderfer, M. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)* (2017), IEEE, pp. 204–211.

[11] Nandy, A., and Biswas, M. Unity ml-agents. In *Neural Networks in Unity*. Springer, 2018, pp. 27–67.

[12] Perez-Liebana, D., Liu, J., Khalifa, A., Gaina, R. D., Togelius, J., and Lucas, S. M. General video game ai: A multitrack framework for evaluating agents, games, and content generation algorithms. *IEEE Transactions on Games 11*, 3 (2019), 195–214.

[13] Quang Tran, D., and Bae, S.-H. Proximal policy optimization through a deep reinforcement learning framework for multiple autonomous vehicles at a non-signalized intersection. *Applied Sciences 10*, 16 (2020), 5722.

[14] Rollings, A., and Adams, E. *Andrew Rollings and Ernest Adams on game design*. New Riders, 2003.

[15] Safadi, F., Fonteneau, R., and Ernst, D. Artificial intelligence in video games: Towards a unified framework. *International Journal of Computer Games Technology 2015* (2015).

[16] Samak, T. V., Samak, C. V., and Kandhasamy, S. Robust behavioral cloning for autonomous vehicles using end-to-end imitation learning. *arXiv preprint arXiv:2010.04767* (2020).

[17] Sander, R. Emergent autonomous racing via multi-agent proximal policy optimization. *Embodied Intelligence* (2020).

[18] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[19] Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954* (2018).

[20] Yannakakis, G. N. Game ai revisited. In *Proceedings of the 9th conference on Computing Frontiers* (2012), pp. 285–292.

[21] Zhu, J., Villareale, J., Javvaji, N., Risi, S., Löwe, M., Weigelt, R., and Harteveld, C. Player-ai interaction: What neural network games reveal about ai as play. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (2021), pp. 1–17.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania

*Email address*: alina.calin@ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXVII**, Number 1, 2022
DOI: 10.24193/subbi.2022.1.03

# ROMANIAN QUESTION ANSWERING USING TRANSFORMER BASED NEURAL NETWORKS

DIACONU BOGDAN-ALEXANDRU AND LÁZÁR-LŐRINCZ BEÁTA

Abstract. Question answering is the task of predicting answers for questions based on a context paragraph. It has become especially important, as the large amounts of textual data available online requires not only gathering information but also the task of findings specific answers to specific questions. In this work, we present experiments evaluated on the XQuAD-ro question answering dataset that has been recently published based on the translation of the SQuAD dataset into Romanian. Our best-performing model, Romanian fine-tuned BERT, achieves an F1 score of 0.80 and an EM score of 0.73. We show that fine-tuning the model with the addition of the Romanian translation slightly increases the evaluation metrics.

## 1. Introduction

Question answering (QA) refers to answering questions based on a context paragraph. The answers are of variable length and contain segments of the provided context paragraph. QA is a natural language processing (NLP) task as it entails the automatic understanding of the text.

The importance of question answering has been discussed for more than two decades, as online information has become widespread and users raised the need for not only gathering information from large collections of documents, but also answering specific questions [7].

The task of QA has been addressed with various approaches such as the ones based on Information Retrieval/Information Extraction (IR/IE), restricted domain systems, or rule-based systems [6]. Most IR based systems are returning a list of top-ranked documents or passages as responses to a query. In the following step, the IE system parses the questions and documents yielding the

Received by the editors: 9 December 2021.

2020 *Mathematics Subject Classification.* 68T07, 68T50.

1998 *CR Categories and Descriptors.* I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *Language models*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *Language parsing and understanding*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing – *Text analysis* .

*Key words and phrases.* question answering, deep learning, Transformer, Romanian.

interpretation of each word, using several resources like Named Entity Tagging, Template Element, Template relation, Correlated Element, and General Element. The limitation of this system is the fact that it can only answer yes/no type of questions and wh-type of questions (such as when, where, what, etc.). Restricted domain systems, as the name suggests, restrict the domain of questions and the size of the knowledge base. In this system, a question is linguistically inspected by the Heart of Gold architecture. The semantic representations are then interpreted, and a question object that contains a proto query is produced. From this, an instance of a specific database or ontology query is created. An answer object is generated from the result(s) returned by the queried knowledge source. This object forms the foundation for subsequent natural language answer generation. Rule-based systems are an extension of the IR-based QA systems. For each type of question, it generates rules for the semantic classes like who, when, what, where, and why type questions. Rule-Based QA systems initiate parse notations and create training cases and test cases throughout the semantic model.

However, more recently the state of the art results are achieved with neural network models, especially with the fully attention-based Transformer [12] model. This neural architecture is commonly applied to NLP tasks, as it is capable of modeling long-range dependencies. The unidirectionality of the standard language Transformer models limits the options for the architectures used at pre-training. BERT (Bidirectional Encoder Representations from Transformers) is a model proposed by [4] with the purpose of reducing this unidirectionality by using a "masked language model" (MLM) for pre-training. The MLM objective allows the representation to analyze the context both to the left and the right, which enables the pre-training of the deep bidirectional Transformer.

Question answering models were proposed for several languages as a result of the availability of datasets that provide training data for these models. However, for under-resourced languages such as Romanian, to the best of our knowledge, the first baseline model for QA is described in [5]. In our work, we aim to analyze a model for Romanian QA for the newly introduced dataset by [5]. The rest of the paper is structured as follows: Section 2 presents the related work, Section 3 details the method describing the dataset and training architectures, in Section 4 the experiments and results are presented, and conclusions are summarized in Section 5.

## 2. Related work

Artetxe et al. [1] introduced a new Cross-lingual Question Answering Dataset (XQuAD) for a better understanding of the cross-lingual generalization ability of the models described in the paper. The XQuAD dataset includes the translation of the paragraphs, questions, and answers from the SQuAD v1.1 dataset [11] into ten languages. The authors also showed that neither a shared vocabulary nor joint pre-training is necessary for multilingual models.

In the work of Xue et al. [16], a new token-free, byte-to-byte pre-trained model is proposed. The authors compared the new model, ByT5 with another model that uses the T5 [10] framework, mT5 (the multilingual variant of a T5 architecture), introduced by Xue et al. [17]. The T5 is a unified framework that converts all text-based language problems into a text-to-text format. These models were tested on multiple tasks included in the GLUE [14] or SuperGLUE [13] benchmarks, as well as on a subset of tasks included in the Cross-lingual TRansfer Evaluation of Multilingual Encoders (XTREME) benchmark [8]. The XTREME multi-task benchmark is proposed for evaluating multilingual representations through their cross-lingual generalization competencies for 40 languages and 9 tasks building upon existing benchmark datasets such as XQuAD, XNLI [3], TyDi QA [2] and many others. The results favor ByT5 on small models and mT5 on large models for the XQuAD dataset, the best scores are of the mT5 with an F1 of 85.2 and Exact Match (EM) of 71.3.

Adrian et al. [9] developed a system used in the Question Answering competition QA@CLEF for the ResPubliQA track in 2010. For their corpus, the JRC-Acquis and Europarl corpora were indexed, both of them containing documents in XML format. Their question analyzer performed 5 tasks: Noun Phrase chunking and Named Entity extraction, question focus identification, question type inferring, answer type identification, and identification of the keywords of the sentence. The output of these tasks was then used to help the following components: The Index Creation, Information Retrieval, and the Answer Extractor. As for the results, they obtained an accuracy of 55% on the Romanian language, 46% on English, and 30% on French. Our approach is different from the one described in [9] as we use Transformers and a different dataset.

Dumitrescu et al. [5] presents three new datasets: the Semantic Textual Similarity (RO-STS), Question Answering (XQuAD-ro), and Language Modeling (Wiki-ro) datasets. Together with five already existing datasets on the Romanian language, the authors published an open-source benchmark and

leaderboard platform for NLP tasks on Romanian language[1]. The benchmark comprises ten tasks including text categorization by topic, question answering, sentiment analysis, etc. Each task is associated with baseline results, for Question Answering, the mBERT and XLM-R Large models from [1] were used, where the XLM-R Large achieves an F1 score of 83.56.

As the Romanian component of the XQuAD dataset[2] is newly introduced, to the best of our knowledge, notable work has not been published on the XQuAD-ro.

## 3. Method overview

3.1. **Dataset.** The XQuAD is a benchmark dataset for evaluating cross-lingual question answering performance. It consists of an English subset of 240 paragraphs and 1190 question-answer pairs from the development set of SQuAD v1.1 [11]. The XQuAD also contains the subset's translation in eleven languages: Spanish, German, Greek, Russian, Turkish, Arabic, Vietnamese, Thai, Chinese, Hindi, and the most recently added one, Romanian.

All files are in json format following the SQuAD dataset format. Every paragraph consists of one title and a multitude of lists of questions and answers related to the given context. The questions have a unique id and a text. The answers have the text and also the position in the context where the answer starts. Finally, context is represented by a text consisting of multiple sentences related to the topic of the title. Table 1 shows an example in Romanian of how the data is structured in the XQuAD dataset.

3.2. **Training architecture.** Because of time constraints and as well as limited hardware capacity, we decided to use a pre-trained BERT model, fine-tuned on XQuAD like data (before Romanian was added)[3] and further improve it by training on the Romanian language as well. This model has been trained on 104 languages, using 12 attention heads and it has 768 hidden neurons and 110M parameters. The languages have been chosen based on their wikipedia's size. To avoid the overfitting of the model on languages that have less content based on their wikipedia pages, the authors have performed an exponentially smoothed weighting of the data during pre-training data creation. For tokenization, they used a 110k shared WordPiece vocabulary [15]. The word counts were weighted the same way as the data, so low-resource languages were upweighted by a factor. Moreover, the model was fine-tuned on a dataset created by using data augmentation techniques (scraping, neural machine translation, etc.) to obtain more samples from the XQuAD dataset.

[1] `https://lirobenchmark.github.io/` accessed in August 2021

[2] `https://github.com/deepmind/xquad` accessed in august 2021

[3] `https://huggingface.co/mrm8488/bert-multi-cased-finetuned-xquadv1`

| Context | Questions | Answer text | Answer start |
|---|---|---|---|
| Apărarea Panthers a cedat doar 308 puncte, clasându-se pe locul șase din ligă, în timp ce au dominat NFL la interceptări, în număr de 24 și s-au putut lăuda cu patru selecții la Pro Bowl. | Câte interceptări a avut apărarea Panthers în sezonul 2015? | 24 | 134 |
| Jucătorul principal al apărării la Pro Bowl, Kawann Short, a condus echipa la numărul de sack-uri cu 11, forțând și trei fumble-uri și recuperând două. | Cine a avut cele mai multe sack-uri în echipa Panthers? | Kawann Short | 233 |

TABLE 1. A sample of the context, questions and answers that can be found in XQuAD-ro.

This increased the size of the dataset from a total of 13,090 question-answer pairs to 58,000 samples. We will refer to this model in the experiments section as XQuAD fine-tuned BERT.

Finally, our contribution to the model consisted in: splitting the dataset, training the model on the new Romanian data with the same number of attention heads and hidden neurons, respectively, and searching for the optimal training arguments. The dataset explained in Section 3.1 has been randomly shuffled and split into 952 question-answer pairs for training (80%), 119 for evaluation (10%), and 119 for testing (10%).

The model trained on the dataset presented above for 10 epochs with a checkpoint on every epoch. Afterward, the best model based on the performance of the F1 score on the evaluation set has been selected. Due to hardware limitations, we used 6 batches to train upon. We used a weight decay of 0.01 and 500 steps of warm-up. In the experiments section, we will refer to this model as Romanian fine-tuned BERT.

## 4. Experiments and Results

The two most dominant metrics used in question answering tasks are the F1 and EM scores. F1 is calculated by computing the harmonic mean of the precision and recall. Precision is the number of shared words between the prediction and the ground truth divided by the total number of words in the prediction. Recall is the ratio of the number of shared words between the prediction and the ground truth to the total number of words in the ground

truth. Equation 1 represents the formula for calculating the F1 score. EM (Exact Match) is the ratio of the number of predictions, that exactly match the characters of the correct answer to the total number of predictions.

$$F1 = \frac{2}{precision^{-1} + recall^{-1}} \tag{1}$$

We have used the *sklearn*[4] library for calculating F1 and the necessary scores used in its formula. The loading of models, training, and predictions were facilitated by the *transformers* [5] library.

The experiments consisted in computing the accuracy, F1, and EM metrics, the last two being the most important, for both the XQuAD fine-tuned BERT and Romanian fine-tuned BERT models on the test set presented in 3.2. Table 4 presents the results.

Examining the results, there is not much of an improvement from the XQuAD fine-tuned BERT to the Romanian fine-tuned BERT. Most likely, that is caused by the small size of the dataset. Compared to the SQuAD dataset [11] which has 100,000+ question-answer pairs, the XQuAD has only one-hundredth of that amount for one specific language. As consequence, the 952 question-answers pairs are not enough to train the model for more than a few epochs without overfitting.

| Model | Dataset | Accuracy | F1 | EM |
|---|---|---|---|---|
| XQuAD fine-tuned BERT | XQuAD-ro | 0.80 | 0.79 | 0.71 |
| Romanian fine-tuned BERT | XQuAD-ro | 0.80 | 0.80 | 0.73 |

Table 2. Models and their computed metrics

## 5. Conclusions

In this paper we have presented question answering experiments performed on the newly published XQuAD-ro dataset. The first experiment evaluated the model on the Romanian dataset without fine-tuning it on the Romanian language, while the second experiment reports the results after fine-tuning with

[4]https://scikit-learn.org/ accessed in November 2021

[5]https://huggingface.co/transformers/ accessed in November 2021

the additional data. We plan to submit our models to the LiRo benchmark after publication.

Comparing the zero-shot model used with the baseline offered by the XQuAD official repository [6], we got a higher score on the EM metric, 0.71 compared to their best model with 0.69, but we got a lower score on the F1 metric, 0.79 compared to their best model with 0.83.

The low difference in the F1 and EM metrics between the two models is due to the small amount of data that the Romanian language has at its disposal for the Question Answering task. To overcome this barrier, data augmentation techniques could be used to enhance the size of the training set and reduce overfitting. Machine translation could also be used on other datasets for a higher training set. The issue with the latter option is that the accuracy of the model is very dependent on the quality of the translation.

## References

[1] M. Artetxe, S. Ruder, and D. Yogatama. On the Cross-lingual Transferability of Monolingual Representations. *arXiv preprint arXiv:1910.11856*, 2019.

[2] J. H. Clark, E. Choi, M. Collins, D. Garrette, T. Kwiatkowski, V. Nikolaev, and J. Palomaki. TyDi QA: A Benchmark for Information-Seeking Question Answering in Typologically Diverse Languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020.

[3] A. Conneau, G. Lample, R. Rinott, A. Williams, S. R. Bowman, H. Schwenk, and V. Stoyanov. XNLI: Evaluating Cross-lingual Sentence Representations. *arXiv preprint arXiv:1809.05053*, 2018.

[4] Devlin, Jacob and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] S. D. Dumitrescu, P. Rebeja, B. Lorincz, M. Gaman, A. Avram, M. Ilie, A. Pruteanu, A. Stan, L. Rosia, C. Iacobescu, et al. LiRo: Benchmark and leaderboard for Romanian language tasks. 2021.

[6] P. Gupta and V. Gupta. A Survey of Text Question Answering Techniques. *International Journal of Computer Applications*, 53(4), 2012.

[7] L. Hirschman and R. Gaizauskas. Natural Language Question Answering: The View from Here. *natural language engineering*, 7(4):275–300, 2001.

[8] J. Hu, S. Ruder, A. Siddhant, G. Neubig, O. Firat, and M. Johnson. XTREME: A Massively Multilingual Multi-task Benchmark for Evaluating Cross-lingual Generalization. In *International Conference on Machine Learning*, pages 4411–4421. PMLR, 2020.

[9] A. Iftene, D. Trandabat, M. Husarciuc, and M. A. Moruz. Question Answering on Romanian, English and French Languages. In *CLEF (notebook papers/LABs/workshops)*, 2010.

[10] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[6] https://github.com/deepmind/xquad accessed in November 2021

[11] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, Nov. 2016. Association for Computational Linguistics.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention Is All You Need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[13] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*, 2019.

[14] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[15] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*, 2016.

[16] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, and C. Raffel. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *arXiv preprint arXiv:2105.13626*, 2021.

[17] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu Street, 400084 Cluj-Napoca, Romania

*Email address*: beata.lorincz@ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXVII**, Number 1, 2022
DOI: 10.24193/subbi.2022.1.04

# MUSIC RECOMMENDATIONS BASED ON USER'S MOOD USING CONVOLUTIONAL NEURAL NETWORKS

ANDREI PETRESCU

Abstract. This paper proposes a method for music recommendations using emotions, using deep learning techniques. The method is composed of two modules. The emotion detection module, which utilizes a hybrid architecture involving a Convolutional Neural Network (CNN) and a Reccurent Neural Network using Long-Short Term Memory (LSTM) Cells. We compared individual architectures of CNNs and LSTMs against our hybrid approach, outperforming them during experiments. We evaluated the modules on our own data set, created using Spotify's API and containing 2028 songs from different genres and linguistic families, labeled with valence and arousal values. The model also outperforms other related approaches, however we did not evaluate them on the same data set. The predictions are used by the second module, for which we proposed a simple method of ordering the results based on the similarity to user's input.

## 1. Introduction

As the popularity of streaming services grows each year, a problem is raised when it comes on which songs (besides the one he saves) should be delivered to the final user. The focus on these services is to provide as much content as possible for a large audience. The recommendations given by them are based on user's history of liked songs, genres, new songs which may be of interest for him etc.

Received by the editors: 10 October 2021.

2010 *Mathematics Subject Classification.* 68T45.

1998 *CR Categories and Descriptors.* I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis – *Object recognition*; I.2.6 [**Artificial Intelligence**]: Learning – *Connectionism and neural nets*; I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding – *Intensity, color, photometry, and thresholding* .

*Key words and phrases.* mood, emotion, valence, energy, convolutional neural network, recurrent neural networks, long-short term memory, hybrid, regression, classification.

The task of music emotion recognition has been an area of interest for many years. This subject was tackled from different perspectives. Music may be annotated with emotion names (labels), or by expressing emotions using continuous values. Most *machine learning* (ML) methods consider features such as: pitch, beat, tempo, rhythm, melody or harmony. These were successfully utilized as inputs for Support Vector Machines and Naive Bayes models [15]. However, traditional machine learning techniques under perform when compared to deep learning methods. When working with high-dimensional data, machine learning methods are usually insufficient to learn more complex functions. For the task at hand, CNNs will be employed to extract abstract features from two-dimensional data, in the form of audio spectrograms [3]. These features are further utilized in Deep Neural Network architectures to predict the results. In most studies, deep learning approaches outperform traditional machine learning models [1].

As mentioned, the task of emotion recognition may be viewed as a classification problem. However, by using Russel's circumplex model of affect [14], we can define emotions using two continuous measures: valence and arousal (energy). This model allows us to reformulate it as a regression problem. Valence is an indicator of "happiness", which measures the positivity of an emotion. Arousal or energy measures the intensity of that certain emotion.

In this paper, we propose a hybrid deep learning architecture for a recommendation system composed of two parts (modules): an emotion detection module and a playlist generator. Using deep learning methods, we emphasised the task of emotion detection, by creating and comparing the performance of three neural network architectures. We first compared the performance of Convolutional Neural Network and Recurrent Neural Networks (RNNs). This approach was successfully used before for classification tasks in music recommendations systems based on genre [13], but, judging by our experiments, these additionally perform well on regression tasks. The proposed CNN-LSTM (Long-Short Term Memory) hybrid network achieves comparable results to state-of-the-art approaches, having the potential to outperform them. In addition, we describe an easy method to generate a playlist based on user's input, by searching for the closest items to his emotion.

We organized the rest of the paper as follows. Section 2 describes the related work utilized in creating and experimenting upon deep learning methods for solving the task at hand. The methodology for representing the data (sound and emotions) and the experimented neural network architectures are detailed

in Section 3. We also discuss about the metrics used for the evaluation and playlist generation. Section 4 contains details about the created data set for our experiments and the results and comparison to related work. Finally, Section 5 contains the conclusions and possible future work.

## 2. Related work

During the 20th century, music studies emerged with the work of Kate Henver [5], who succeeds to unveil a correlation between emotions and music characteristics. Taking in consideration the work done for audio-based mood detection, in the last 20 years, different approaches have been developed. Such works include Tao Li and Mitsumori Ogihara's [7], in which they use audio features as timbre, rhythm and pitch to detect emotions in music, or Geoffroy Petter's [11] Support Vector Machine. He used Mel-Frequency Cepstral Coefficients as an input for his SVM.

As advancements in deep learning technology occurred, new models emerged based on fewer feature engineering. Music Information Retrieval Evaluation eXchange (MIREX) competition has unveiled the evolution of state of the art. Thomas Lidy and Alexander Schindler's work [8] shows the potential of audio-based models, using CNNs. However, these approaches are based on labeling music by their emotions. Two state-of-the-art methods using CNNs, that use the same data representation as our work, consist in the works of Delbouys et al. [3] and Bhattarai et al. [1]. These methods implement a fully-connected network as the final layer, which will output the prediction.

Recent music recommendation systems utilize deep learning in order to identify musical content [16]. Raju et al. utilize a hybrid network using a CNN and an LSTM module for music genre classification [13]. This method inspired our approach, which was further compared with other similar works that implement this type of network for emotion recognition. A very similar method was implemented by Malik et al. [10], who uses a bidirectional-GRU module, instead of consecutive LSTM layers.

## 3. Methodology

This section will approach in multiple subsections the methodology used in elaborating emotion detection method. First, it is discussed how the data is represented in particularly the audio signal and the emotion model utilized. In the following subsections, we will present the deep learning methods, which

form the hybrid architecture. These methods will also be individually implemented for comparison. The final model consists in a combination of a CNN module and a RNN using LSTM cells. The combination of these models was created setting the goal to achieve better performance than state-of-the art and to outperform the individual models.

## 3.1. **Data Representation.**

3.1.1. *Emotion Representation.* Russell's valence-arousal (V-A) model [14] is one of the most widely used models for describing emotions. Because people experience interactions differently, this approach seeks to express emotions objectively in a way that mere labels can not. Emotions are represented on a two-dimensional space in this model. The positive effect of an emotion is represented by valence. Valence levels can be interpreted as being negative or positive, or low or high, depending on the scale employed. Happiness, for example, might be classified as a positive or high-valence emotion. The intensity of an emotion is represented by arousal. Anger, for example, is a powerful emotion with a high arousal value. Therefore, the scale utilized determines the representation. Valence and arousal are given values between 0 and 1 in our experiments. The V-A model describes happiness as having high values, near to 1, for both components.

3.1.2. *Mel-Spectrograms.* The audio signal may be represented in the form of an image in the form of a spectogram. Using spectrograms, we can take advantage of CNN's performance on multidimensional data [9]. Applying the Short-Time Fourier Transform [9], we obtain the spectrogram.

Inspired by Delbouys et al [3], we took into consideration that the human ear cannot differentiate sounds of low or high frequencies. Beginning from a frequency $f$, we can re-scale the values to Mel Scale. The converted value $m$ is obtained applying the following formula [6]:

$$m = 2595 * \log(1 + \frac{f}{500}) \tag{1}$$

After converting all the frequencies using Formula. 1, we will obtain the final mel-spectrogram. The output will be saved as a 128 x 128 px grayscale image as shwon in Figure. 1.

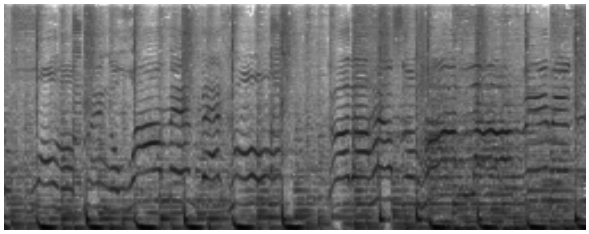


FIGURE 1. Example of a mel-spectogram

3.2. **Proposed Architecture.** In order to create a recommendation system based on emotions for music, we divided it into two parts. The first part will detect an emotion from a musical piece and the second will deliver the recommendations based on a user's input (his emotion using the valence-arousal model described in the Section. 3.1.1). This paper focuses on the emotion detection, for which we created a hybrid neural network, that may outperform state-of-the-art approaches. In the following sections we will present the architecture, the performance metrics used for the evaluation and a proposal for generating recommendations.

3.2.1. *Convolutional Neural Network.* These types of networks are known for the capability of extracting abstract features from multidimensional data. Beginning from a 128 x 128 matrix, representing the mel-spectrogram extract the features, further processed. Each convolution layer multiplies parts of the last layer's output. The result may, or may not outline the important information contained by the input matrix. Another type of layer, that CNNs use, is the pooling layer. This study utilizes the Max-Pooling layers to down-sample the the output of convolutions. Max-Pooling layers downsizes the input using the maximum values from a stride. A stride is a portion from a matrix having a fixed size [1].

The CNN module (Figure 2), utilized for the hybrid and individual architectures, will use multiple convolution and pooling layers. This architecture was inspired from the works of Liu et al. [9], that used a CNN architecture for music emotion recognition. Our CNN alternates between a convolution layer and a max-pooling layer as shown in Figure 2. Between them, we use a batch normalization layer to keep values under control and avoid over fitting. For individual analysis during the experiments, we connected the module to

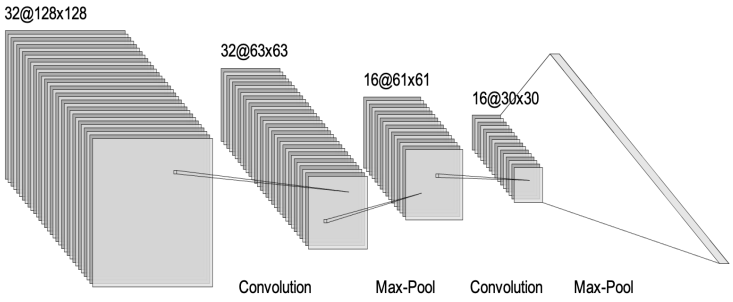



FIGURE 2. CNN module

a fully connected Deep Neural Network (DNN) (Figure 3). Its final layer has only two neurons computing the values of valence and energy.

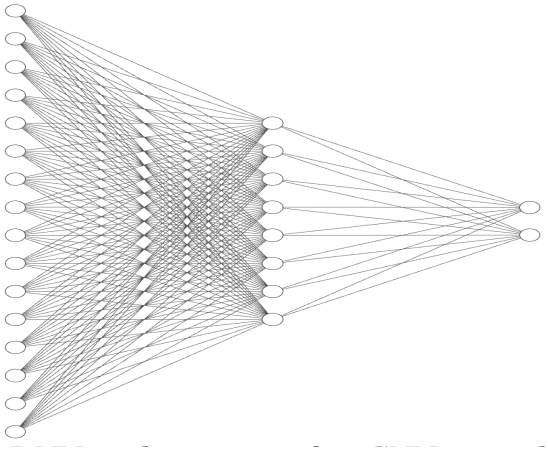


FIGURE 3. DNN module

3.2.2. *LSTM-RNNs.* The second method studied involves RNNs. This type of architecture has proven its performance on solving emotion recognition problems. However, traditional RNNs suffer from what is known as "the vanishing/exploding gradient" problem. This prevents RNNs from further learning. To avoid this problem, we use LSTM Cells, described by formulas (2) and (3) [4]:

$$c_t = f_t \odot \ c_{t-1} + i_t \odot \widetilde{c}_t \tag{2}$$

$$\widetilde{c}_t = g(W_c * x_t + U_c * \ h_{t-1} + b_c) \tag{3}$$

$$h_t = o_t \odot g(c_t) \tag{4}$$

In Formulas (2) and (3), $t$ is the current timestamp, $c_t$ is the current cell value, $\widetilde{c}_t$ is the proposed cell, $h_t$ is the hidden state, g is the activation function. $W$, $U$, $b$ are the prior computed parameters, which are updated during backpropagation. During the computations, LSTMs use three types of signal gates: $f_t$ forget gate, $o_t$ output gate, $i_t$ input gate. The formulas for computing the gates are [4]:

$$i_t = \sigma(W_i * x_t + U_i * \ h_{t-1} + b_i) \tag{5}$$

$$f_t = \sigma\left(W_f * x_t + U_f * h_{t-1} + b_f\right) \tag{6}$$

$$o_t = \sigma(W_o * x_t + U_o * \ h_{t-1} + b_o) \tag{7}$$

In Formulas (5), (6) and (7), the parameters $W_{i,f,o}$, $U_{i,f,o}, b_{i,f,o}$ correspond to each gate for signal computations and are updated using backpropagation during the training stage. Our two-dimensional input, may be divided in fixed time-steps lengths. The module uses two LSTM layers with 40 and 2 units. For individual comparison with the hybrid module, the module is connected to a Dense layer, that will compute the output.

3.2.3. *The proposed Hybrid Network.* In the previous subsections we described the individual modules, which combined will form the hybrid architecture. The network begins with the CNN, reshaping the input by dividing it into time steps. We used the following formula for shaping the input:

$$inputShape_{Hybrid} = (no.\ of\ images,\ no.\ of\ t.\ steps,\ 128, \frac{128}{no.\ of\ t.\ steps},\ 1) \tag{8}$$

The first parameter from Formula (8) represents the number of images processed by the network. For each image the network outputs one prediction.

The second parameter is the number of time steps used. Controlling this parameter will affect the performance of the model. Last three parameters refer the image's sizes. The output of the CNN module is flattened and directed to the LSTM module, which will predict the emotions.

3.2.4. *Performance Evaluation.* We compared our model with state-of-the-art approaches that utilize CNNs or RNNs to detect emotions from music. These approaches use different performance metrics specific for a regression model. Therefore, we evaluated our models using all performance metrics met in the compared papers. The performance metrics are:

- *Mean Squared Error* (MSE)

$$MSE = \frac{1}{n} * \sum_{i=1}^{n} \left(Y_i - \widetilde{Y_i}\right)^2 \tag{9}$$

- *Root Mean Squared Error* (RMSE)

$$RMSE = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} \left(Y_i - \widetilde{Y_i}\right)^2} \tag{10}$$

- *Mean Absolute Error* (MAE)

$$MAE = \frac{1}{n} * \sum_{i=1}^{n} |Y_i - \widetilde{Y_i}| \tag{11}$$

- $R^2$ score ($R^2$)

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(Y_i - \widetilde{Y_i}\right)^2}{\sum_{i=1}^{n} \left(Y_i - \overline{Y}\right)^2} \tag{12}$$

In Formulas (9), (10), (11) and (12) the expected value for the i-th input is denoted by $Y_i$. $\widetilde{Y_i}$ represents the predicted value and $\overline{Y_i}$ is the mean of the expected values. To obtain a better performance, our models require, on one hand to minimize MSE, RMSE and MAE values, on the other hand to maximize the $R^2$ score.

3.2.5. *Recommendations.* After using the model for predicting the valence and arousal values for all input images, we can generate a list (playlist) of music based on an user's mood. The goal is to recommend music as close as possible to user's emotions. A user may characterize his current mood using the

valence-arousal model described in Section. 3.1.1. Each song is now characterized by these values too, therefore we need to choose the most similar songs for an user's mood. The values range from 0 to 1 and the data can be represented in a two-dimensional space. For computing the similarity we can use a distance based metric. An example of such a metric is the Euclidian distance computed using the following formula:

$$D_i = \sqrt{(v_i - v_u)^2 + (a_i - a_u)^2} \tag{13}$$

In Formula (13), $v_i$ and $a_i$ are the valence and arousal values corresponding to a song $i$. $v_u$, $a_u$ are the valence and arousal values from an user's input. For each song we will compute this distance $D_i$ and sort them from the lowest to highest score. From the obtained list, we can extract first $n$ songs, which will form the playlist for a user.

## 4. Experimental Evaluation

This section presents the conducted experiments and the obtained results. For testing the proposed methods, we created and utilized our own data set containing the generated mel-spectrograms and corresponding values for valence and arousal. In the following subsections we discuss the results for the individual architectures (CNN and LSTM) and the novel hybrid approach for solving this task.

4.1. **Data Set.** Using Spotify's API [2], we collected the necessary audio files. This API provides a 30 seconds mpeg-3 audio clip as a preview for the entire song. This audio clip is the most meaningful and listened part of the song, according to API documentation [2]. For creating this data set we collected audio previews from the following genres: Jazz, Rock, Classical, Hip-Hop, Folk and Electronic. The data set may include songs from one or more sub genres from the ones mentioned above [12]. The music belongs to the following linguistic families: Latin, Slavic, Germanic, Indo-Iranian and Japanic. Another characteristic is that the data set contains instrumental and non instrumental songs, because we require that our methods may focus on melody and not on verses. The API also provides the valence and arousal (referred as energy) values for each song.

The final data set [12] to be used during the training and testing phases is composed of the generated mel-spectrograms (in the form of a gray scale

image) from the collected audio clips and contains 2028 entries. To each image, we associate the values for emotion. The mean values for valence and arousal are 0.476 and 0.526. The variance values for both features are 0.064 and 0.071.

4.2. **Experimental Setup.** As mentioned before, we trained and compared the performance of the individual modules and the hybrid network in solving the same task. We used k-fold cross-validation in order to detecting overfitting. We chose $k = 4$ for our implementation and each model is trained for 5000 epochs. Multiple activation function were used in our models, however the last layer of neurons is activated using the sigmoid activation. In case of the CNN module, we used ReLU activation functions. For the LSTM module we utilized hyperbolic tangent activation and sigmoid activation for recurrent activation. The rest of hyperparameter values are presented in Table. 1. For the hybrid network we took into consideration the possible effect of sequence lengths. We tested the performance on lengths of 16 and 32 time steps.

| **Hyperparameter** | **Value** |
|---|---|
| Dropout $\beta$ | 0.2 |
| Learning Rate | 0.001 |
| Data Set Split Ratio | 80/20 (training/test) |
| Batch Size | 50 |
| No. of Epochs | 3000 |

TABLE 1. Hyperparmeter values

4.3. **Results and Discussion.** The first experiment involves testing different time sequence lengths to achieve a better performance for the hybrid model. We will present the average results for the train and test data sets, but, in order to compare our approach with the related work, we considered computed the results for the entire data set (Table 4). The obtained results (Table 2) point out that a smaller time sequence length improves the performance, when applied on the testing data set. The performance increases for the training data as well, but, for the most cases, it is similar. The length's decrease has the greatest impact on the coefficient of determination ($R^2$ Score), which increases for the testing data. If we compare the results to the individual modules, the hybrid network obtains the best performance (Table 3). However, the CNN module achieves close results and obtains a better RMSE score for the testing data. The LSTM module performs the worst out of all three and

has a tendency of over fitting. Therefore, the most impact on the overall performance of the hybrid network it is due to the CNN module. Although, the extracted abstract features are better utilized by the LSTM module in order to predict the emotions, than the Deep Neural Network utilized for the individual CNN. A graphical illustration of the performances of the considered ML models on the testing data is illustrated in Figure 4.

| **Length** | **MSE** | | **RMSE** | | **MAE** | | **$R^2$ Score** | |
|---|---|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test | Train | Test |
| **16** | **0.001** | **0.031** | **0.027** | **0.178** | **0.022** | **0.139** | **0.996** | **0.413** |
| 32 | 0.001 | 0.038 | 0.028 | 0.196 | 0.022 | 0.152 | 0.995 | 0.292 |

TABLE 2. Performance results of the hybrid network for different time sequence lengths

| **Architecture** | **MSE** | | **RMSE** | | **MAE** | | **$R^2$ Score** | |
|---|---|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test | Train | Test |
| CNN | 0.011 | 0.035 | 0.105 | 0.189 | 0.081 | 0.153 | 0.931 | 0.323 |
| LSTM | 0.007 | 0.061 | 0.029 | 0.247 | **0.020** | 0.195 | 0.854 | 0.234 |
| CNN+LSTM | **0.001** | **0.031** | **0.027** | **0.178** | 0.022 | **0.139** | **0.996** | **0.413** |

TABLE 3. Performance results for all the studied methods

| **Architecture** | **$R^2$ Score** |
|---|---|
| CNN | 0.849 |
| LSTM | 0.798 |
| CNN+LSTM | **0.901** |

TABLE 4. Performance results for the entire data set

4.4. **Comparison to Related Work.** Throughout the literature, the task of predicting the mood from music was considered a classification problem. Data was labeled with different emotion names, which, in our opinion, simplifies the spectrum of existing emotions. By abstracting an emotion and quantifying it, we used the valence/energy model, which can accommodate a larger scale of emotions, without diminishing their complexity.

This approach was utilized before by other authors such as Delbouys et al. [3], for predicting emotions using CNNs. Tables 5, 6 and 7 compare our results

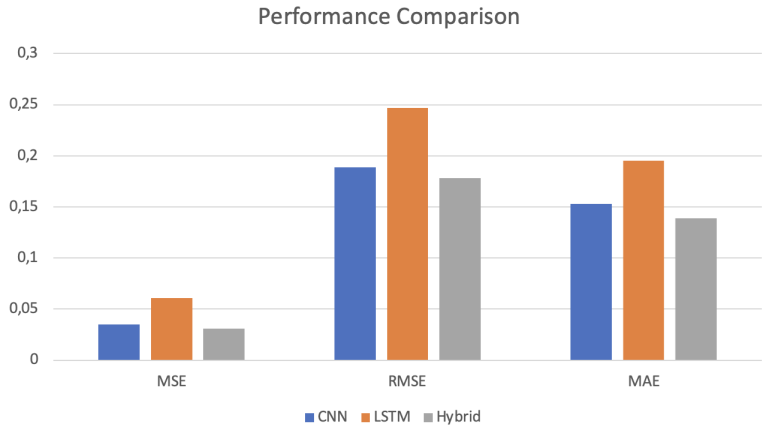



FIGURE 4. Comparison between the performance of studied methods (Table 3) on the testing data

to three similar approaches, those of Bhattarai et al. [1], aforementioned Delbouys et al. [3] and Malik et al. [10]. The last approach is the most similar to our hybrid network. Instead of two stacked LSTM layers, the authors use a bidirectional GRU network. We wanted to compare the impact on performance of a simpler recurrent network to a more complex one, in the form of Malik et al.'s approach [10]. The works compared in our study evaluate their models using parts (train/test) or entire data sets.

| **Approach** | **Dataset size** | **$R^2$ Score** |
|---|---|---|
| Bhattarai et al. [1] | 1000 x 96 x 1360 | 85.61% |
| CNN | 2028 x 128 x 128 | 84.90% |
| LSTM | | 79.80% |
| CNN + LSTM | | **90.10%** |

TABLE 5. Comparison between approaches using $R^2$ for the entire dataset

As shown in Tables 5 and 6, our hybrid approach achieves an $R^2$ score of 90.10% applied on the entire data set an a score of 41.30% for testing data. These results outperform the the proposed methods of Bhattarai et al. [1] and Delbouys et al. [3], that utilize CNNs to detect emotions. In our individual experiments using only CNNs, we achieved close results to

| **Approach** | **Dataset size** | **$R^2$ Score** |
|---|---|---|
| Delbouys et al. [3] | 18644 x 40 x 1292 | 24.3% |
| CNN | 2028 x 128 x 128 | **31.70%** |
| LSTM | | 23.40% |
| CNN + LSTM | | **41.30%** |

TABLE 6. Comparison between approaches using $R^2$ for the test data

| **Approach** | **Dataset size** | **RMSE** |
|---|---|---|
| Malik et al. [10] | 431 x 60 x 260 | 0.255 |
| CNN | 2028 x 128 x 128 | 0.189 |
| LSTM | | 0.247 |
| CNN + LSTM | | **0.178** |

TABLE 7. Comparison between approaches using RMSE for the test data

those approaches, even outperforming Delbouy's method. When comparing the two similar hybrid networks, our method obtained a RMSE value of 0.178, outperforming the approach proposed by Malik et al. [10]. However, even if we obtained better results than the compared related works, we cannot state the superiority of our method, because we did not experiment upon the same data sets. The data sets used by the authors are privately owned and cannot be accessed without permission. Figure 5 visually represents the comparison between our proposed hybrid method (CNN+LSTM) and the approaches of Bhattarai et al. [1], Delbouys et al. [3] and Malik et al. [10], as represented in Table 5, Table 6 and Table 7.

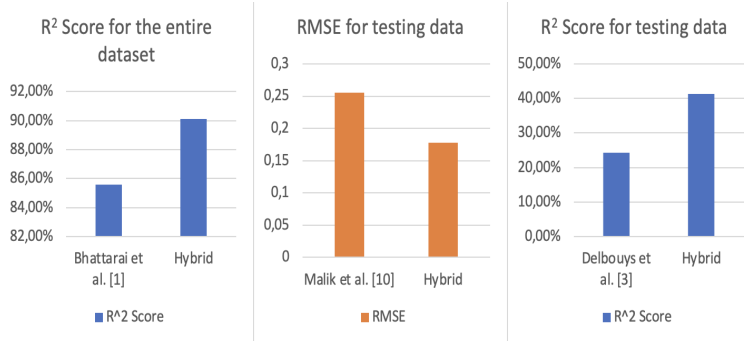



FIGURE 5. Performance comparison between the proposed hybrid method and the approaches of Bhattarai et al. [1], Delbouys et al. [3] and Malik et al. [10] (Table 5, Table 6, Table 7)

## 5. Conclusions and future work

In this study, we proposed a method for generating music recommendations based on emotions. Our approach consists of two modules: an emotion recognition module and a recommendation generator based on an user's input. First, we introduced the valence and arousal method created by Russell [14] for annotating data. Then, we studied different deep learning methods for emotion detection, beginning from the mel-spectrogram of a song. These methods are using CNNs, RNNs with LSTM cells and a hybrid network combining a CNN and an LSTM module. Even if the paper focuses on emotion recognition, we detailed a method to generate a personalized playlist, having the music labeled with valence and arousal values. This method searches for k-closest songs to a user's emotion.

We experimented and evaluated all the studied methods and concluded that combining CNN's ability to extract abstract features from multidimensional inputs and LSTM's performance on sentiment analysis tasks, we can obtain a model that outperforms both of them, when used individually. However, from our experiments, it was indicated that the CNN module has the most impact on the performance. Our work was compared to other similar approaches that utilize CNN for music emotion recognition. Based on the results, our method achieves better performance than the compared models. The experiments were not conducted on the same data sets, therefore we are not able to state the superiority of our approach.

Even if the hybrid network achieves better performance, we need to take int account the added computational expense. Using only a CNN module for this task, achieves very close performance and, in future work, we need to further experiment on different architectures, which may be able to achieve better results. However, considering the work of Malik et al. [10], we may further improve the performance using bidirectional RNN layers instead of sequential LSTM layers. We shall further analyze other hybrid architectures involving CNNs and forms of RNNs, that may be able to improve the prediction results.

## References


[1] Bhattarai, B., and Lee, J. Automatic music mood detection using transfer learning and multilayer perceptron. *International Journal of Fuzzy Logic and Intelligent Systems 19*, 2 (2019), 88–96.

[2] Clifton, A., Pappu, A., Reddy, S., Yu, Y., Karlgren, J., Carterette, B., and Jones, R. The spotify podcast dataset. *arXiv preprint arXiv:2004.04270* (2020), 1–4.

[3] Delbouys, R., Hennequin, R., Piccoli, F., Royo-Letelier, J., and Moussallam, M. Music mood detection based on audio and lyrics with deep neural net. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018* (2018), pp. 370–375.

[4] Dey, R., and Salem, F. M. Gate-variants of gated recurrent unit (GRU) neural networks. In *IEEE 60th International Midwest Symposium on Circuits and Systems, MWSCAS 2017, Boston, MA, USA, August 6-9, 2017* (2017), IEEE, pp. 1597–1600.

[5] Hevner, K. Experimental studies of the elements of expression in music. *The American Journal of Psychology 48*, 2 (1936), 246–268.

[6] Kamm, T., Hermansky, H., and Andreou, A. G. Learning the mel-scale and optimal vtn mapping. In *Center for Language and Speech Processing, Workshop* (1997), pp. 1–8.

[7] Li, T., and Ogihara, M. Detecting emotion in music. *CiteSeer* (2003), 1–3.

[8] Lidy, T., and Schindler, A. Parallel convolutional neural networks for music genre and mood classification. *MIREX2016* (2016), 1–4.

[9] Liu, T., Han, L., Ma, L., and Guo, D. Audio-based deep music emotion recognition. *AIP Conference Proceedings 1967*, 1 (2018), 040021.

[10] Malik, M., Adavanne, S., Drossos, K., Virtanen, T., Ticha, D., and Jarina, R. Stacked convolutional and recurrent neural networks for music emotion recognition. *CoRR abs/1706.02292* (2017).

[11] Peeters, G. A generic training and classification system for mirex08 classification tasks: audio music mood, audio genre, audio artist and audio tag. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR'08)* (2008), Citeseer.

[12] Petrescu, A. Spotify dataset. https://github.com/AndreiPetrescu99/SpotifyDataset.git/, 2022.

[13] Raju, A., R.S, D., Gurang, D., Kirthika, R., and Rubeena, S. Ai based music recommendation system using deep learning algorithms. *IOP Conference Series: Earth and Environmental Science 785* (06 2021), 012013.

[14] Russell, J. A. A circumplex model of affect. *Journal of personality and social psychology 39*, 6 (1980), 1161.

[15] Tan, K., Villarino, M., and Maderazo, C. Automatic music mood recognition using russell's twodimensional valence-arousal space from audio and lyrical data as classified using svm and naïve bayes. *IOP Conference Series: Materials Science and Engineering 482* (03 2019), 012019.

[16] Yang, G. Research on music content recognition and recommendation technology based on deep learning. *Security and Communication Networks 2022* (03 2022), Article ID 7696840.

Department of Computer Science, Babeş-Bolyai University, 1, M. Kogalniceanu Street, 400084, Cluj-Napoca, Romania

*Email address*: andrei.petrescu@stud.ubbcluj.ro

STUDIA UNIV. BABES–BOLYAI, INFORMATICA, Volume **LXVI**, Number 1, 2022
DOI:10.24193/subbi.2022.1.05

# A DYNAMIC APPROACH FOR RAILWAY SEMANTIC SEGMENTATION

ANDREI-ROBERT ALEXANDRESCU AND ALEXANDRU MANOLE

Abstract. Railway semantic segmentation is the task of highlighting rail blades in images taken from the ego-view of the train. Solving this task allows for further image processing on the rails, which can be used for more complex problems such as switch or fault detection. In this paper we approach the railway semantic segmentation using two deep architectures from the U-Net family, U-Net and ResUNet++, using the most comprehensive dataset available at the time of writing from the railway scene, namely RailSem19. We also investigate the effects of image augmentations and different training dataset sizes, as well as the performance of the models on dark images. We have compared our solution to other approaches and obtained competitive results with larger scores.

## 1. Introduction

Railway transportation is one of the most efficient modes of moving people and goods from one location to another [7]. The original train routes, which consisted of a small number of stops connecting one point of interest to another, were employed for industrial purposes. More stations were created to assist railway transit as more enterprises saw it as a viable way of carrying freight and passengers. As a result, there was a greater demand for routes between stations.

While numerous advances in scene understanding for autonomous driving have been made in recent decades, one subject has received little attention: *autonomous trains*. Such systems should require as little human intervention as possible. Although fully-autonomous metro systems exist in some modern cities, smart systems for long distance cargo trains are still to be developed.

Received by the editors: 16 July 2022.

2010 *Mathematics Subject Classification.* 68T10, 68T45.

1998 *CR Categories and Descriptors.* I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis – *Object recognition*; I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding – *Intensity, color, photometry, and thresholding*.

*Key words and phrases.* Binary Semantic Segmentation, Encoder-decoder, Railway blades, Deep Learning.

At first glance, the task of building smart trains appears to be simpler to solve than the problem of smart cars or trucks. Trains have a limited range of motion due to the rails on which they travel, thus most of the autonomy consists of adjusting the speed based on different factors such as rail topology (curves, switches), obstacles or adverse weather conditions. In reality, it may be as difficult to solve the task of railway scene understanding as it is to solve the task for road scene understanding, since there are many different traffic signs and lights located in various places along the rail track.

When building a fully autonomous train, the semantic segmentation of the rails is an important aspect that must be considered. This task can be considered as a subproblem for more complex tasks such as detection of switches [12] or anomalies [11], adapting the speed of movement based on the topology of the rails or smart breaking in case of obstacles. It is critical to build a model that can accurately highlight the rails in an image with as few incorrect pixels as possible. This solution might be used in a safety-critical system where even the tiniest mistake could result in derailment or even crashes.

Currently, the task of rails detection can be solved by using two different approaches [22]. The first one implies using image processing techniques such as image edge detection to search for rail features. The second one consists of using deep convolutional neural networks with powerful semantic segmentation potential. This approach can extract edges, colors or textures of rails in more complex images with multiple rail intersections.

In this paper we propose an intelligent solution to the rails semantic segmentation problem using deep neural networks, which leads to better results when compared to the current literature on this problem using the most comprehensive dataset from the rail scene available. Our solution receives as input an image taken from the egocentric point of view of the train containing one or multiple rail tracks. The output is an image of the same size as the input containing white pixels for the rail blades and black pixels for everything else.

The aim of this paper is to answer the following research questions:

- How reliable are the proposed methods for semantic segmentation given a dynamic environment (i.e. the camera on the train)?
- How can we surpass the current state-of-the-art for the rails segmentation problem?

The structure of the paper is the following: Section 2 describes related methods used for solving this task, Section 3 presents the proposed approach for the rails semantic segmentation task and Section 4 describes the experiments and the obtained results. Section 5 concludes the paper by offering an overview of the work and some future considerations.

## 2. Related Work

The task of rail semantic segmentation has received some attention in the past years.

Wang et al. [24] developed an end-to-end model that combines feature extraction using the ResNet-50 backbone [8], followed by a fully convolutional network for detecting the railroad based on a custom Railroad Segmentation Dataset (RSDS) consisting of 3000 images of size 1920x1080 divided as it follows: 2500 images for training, 200 for validation and 300 for test. They achieve a reasonable inference time of 20FPS (Frames Per Second) by extracting both the rails and their interior area, namely the sleeper. The weights used by the ResNet50 backbone were trained on the ImageNet dataset [18] by performing a fine-tuning process. They obtain a mean IoU of 0.898 and a Dice score (F1 measure) of 0.868.

Zhen Tao et al. [22] use a deep neural network called *RailNet* for extracting the rail lines features from an image. This network is trained to generate the binary segmentation map of the rails, which is then processed together with the orginal image using a line fitting algorithm based on a sliding window technique (two-stage detector). Since the system will be used in real-time situations, the inference time represents a key factor to be taken in consideration. In order to obtain a fast inference time of 74FPS and allow the model to be used on memory-constrained devices, they use *Depthwise Convolutions*, which were initially introduced in [19]. They create a custom dataset called *RAWRail* containing 3000 railroad tracks pictures of size 640x360 in which there may be three different types of tracks: straight, curved to the left or curved to the right, 1000 samples for each type. The images are grouped together with the segmentation mask containing the two parallel rails as the positive class and the background as the negative class. They divided the dataset into training, validation and test sets following the ratio 0.9:0.05:0.05 and manage to obtain an accuracy of 98.6% on the test set.

Zendel et al. [26] have created the largest dataset presently available comprising annotated photos obtained from the egocentric perspective of trains, called RailSem19. They have applied deep learning approaches to solve the semantic segmentation task, employing the FRRN (Full-Resolution Residual Network) architecture, which was pre-trained on the Cityscapes dataset [4] and fine-tuned using 4000 training photos randomly selected from the dataset. Based on the ResNet50 backbone, this architecture comprises of an end-to-end model that combines feature extraction and semantic segmentation. The FRRN architecture comes in two flavors: FRRN A and FRRN B, which differ in terms of the input image size: FRRN A processes images of $256 \times 512$ pixels and FRRN B processes images of $512 \times 1024$ pixels. They have used the FRRN

B version with a $512 \times 512$ input size because it has a larger receptive field, performing better than FRRN A. For the semantic segmentation task, after 60 epochs of training, Zendel et al. have obtained an intersection over union (IoU) of 71.5% for the rails class. The images were resized to $512 \times 512$ and they used a batch size of 2 on a single RTX2080Ti GPU.

Li et al. [15] have also used the RailSem19 dataset for the semantic segmentation task, but on another architecture named *RailNet*. The authors have used a VGG-16 backbone [20] on top of which they have added an Information Aggregation Module (IAM) that builds a relationship between each row and column of pixels from the image to semantically segment the rail blades. The weights of this module are acquired in two ways: by using simple learnable weights (RailNet-LW) that get updated with the gradient descent process or by using attention-based weights (RailNet-AW) that work better with the unbalanced class distribution. They divided the RailSem19 dataset into 5000 images for training and 3500 for validation and managed to obtain a mean IoU of 0.54 and a mean recall of 0.89 on the validation set using the attention-based version. They have resized the images to 160x320 and have used an extended version of the focal loss [1].

Jahan et al. [11] have also used the RailSem19 dataset to perform semantic segmentation on the rails using deep learning architectures. They used a U-Net type architecture in which the feature extractor backbone was changed to either VGG [20] or ResNet [8] in order to increase its performance. The weights of these networks were pre-trained on the ImageNet dataset [18]. They also experimented with two types of loss functions: Weighted Binary Cross-Entropy and Focal Loss. Instead of working with grayscale images, Jahan et al. [11] used RGB images of size $892 \times 596$ with a batch size of 4 on two NVIDIA GeForce GTX 1080 Ti GPUs. They made use of image augmentation techniques to increase the performance of the models by using horizontal flipping, random noise, random brightness and random contrast. They obtain the best mean intersection over union (mIoU) of 52.78% after 46 epochs when training on 8390 images, validating on 60 samples and testing on 50 images.

## 3. Our Approach

In this section we present our approach for the rails semantic segmentation problem using U-Net architectures. We describe the formalization of the task, the chosen architectures, the dataset and the loss functions used for experimentation.

3.1. **Formalism.** We define a two-dimensional image as a bidimensional matrix with $r$ rows and $c$ columns with topology $I = \{1, ..., r\} \times \{1, ..., c\}$. Define $img : I \to O$, where $O$ has one of the following forms based on the image type:

- RGB image: $O = \{0, ..., 255\}^3$;
- grayscale image: $O = \{0, ..., 255\}$;
- binary image: $O = \{0, 1\}$.

Therefore, $img(i, j) = o$, $o \in O$, $i \in \{1, ..., r\}$, $j \in \{1, ..., c\}$, where $(i, j)$ is a pair of integers denoting the coordinates of the image and $o$ represents its value at that position.

We define the segmentation mask or the ground truth as $Y_{GT} : I \to \{0, ..., K-1\}$ where $K$ represents the number of types of objects considered for segmentation or the number of different segments considered. Therefore, $Y_{GT}(i, j) = k, k \in \{0, ..., K-1\}$. Similarly, the prediction given by the segmentation model can be defined as $Y_p : I \to \{0, ..., K-1\}, Y_p(i, j) = k, k \in \{0, ..., K-1\}$.

The semantic segmentation model considered in this paper can be formalized as an algorithm that takes as input an image $img$ and outputs an image $mask$ where $img, mask : I \to O$.

3.2. **Architectures.** For the rails segmentation problem we have chosen models from the U-Net family, which feature an encoder-decoder architecture with skip connections between distanced layers. Although these types of architectures were designed to solve the semantic segmentation task for medical images, many studies have shown how well they work for other tasks as well [2, 14]. In our experiments we have considered two model architectures: U-Net [17] and ResUNet++ [13].

3.2.1. *U-Net.* This architecture has a U-like structure formed of a contracting path and an expansive path [17]. The contracting path, otherwise known as the downward or encoder path, is used to learn what features are present in the image, while the expansive path, known as the decoder path, is used to distinguish where the learnt features are located in the image. Between the two parts of the network, skip-connections are used in order to concatenate depthwise information from the downward path to the expansive path. In 2015, the ISBI cell tracking challenge[1] was won by the U-Net architecture, showing state-of-the-art performance at that time.

3.2.2. *ResUNet++.* This architecture, introduced in 2019 builds on top of U-Net [13]. It adds the following: Squeeze and Excite blocks, Atrous Spatial Pyramidal Pooling (ASPP), and attention blocks. The Squeeze and Excite blocks [9] are used to recalibrate channel-wise feature responses by explicitly modelling interdependencies between channels. Atrous Spatial Pyramidal Pooling [3] is used to capture contextual information at various scales. ASPP

[1]https://biomedicalimaging.org/2015/program/isbi-challenges/.

acts as a bridge between the encoder and the decoder part of the architecture. Attention Units are used to enhance the weights of some layers by learning on what parts of the image to focus more, i.e. to pay more attention to.

All of the mentioned architectures work with squared input images for which we choose $512 \times 512$ as size representation.

3.3. **Dataset.** RailSem19 [26] is the most extensive dataset from the railway scene at the time of writing, and we have used it in our study to train models that solve the rail blade semantic segmentation challenge. It consists of 8500 photos captured from the train's ego-view, having the size of $1920 \times 1080$ pixels. The images were captured in a variety of weather, lighting, and seasons in 38 different countries. Ground-truth masks for the rails segmentation procedure and bounding boxes for various elements from the railway scene are included in the samples. The dataset is imbalanced from the perspective of rails:background pixels ratio. For each pixel annotated as rail, there are $\approx 37$ pixels annotated as background.

3.4. **Loss Functions.** We have tried to use different loss functions during the training step of our approach: *Binary Cross-Entropy*, *Weighted Binary Cross-Entropy*, *Tversky similarity index* and *Focal Tversky Loss function.* In the following we present the definitions of these functions.

Let $y$ be the ground-truth and $\hat{y}$ the value predicted by the model. For the segmentation problem with two classes, *rails* and *background*, the Binary Cross-Entropy [6] loss function is defined as:

$$BCE(y, \hat{y}) = -(ylog(\hat{y}) + (1 - y)log(1 - \hat{y})). \tag{1}$$

Since the RailSem19 dataset used in our study is imbalanced, the Weighted Binary Cross-Entropy version of the loss was also considered:

$$WBCE(y, \hat{y}) = -(w * ylog(\hat{y}) + (1 - y)log(1 - \hat{y})), \tag{2}$$

where $w$ is the weight represented by the ratio between background and rails pixels. This way, the loss gives more weight to the positive class (i.e., rails class in our study) when y=1, thus leading to a higher value of the loss function in cases the predicted value $\hat{y}$ is off. This allows the optimizer to improve the model predictions for the positive class.

The Tversky similarity index [23] was used to balance False Positives and False Negatives. This index is a generalization of 4.1, expressed as:

$$TI_c = \frac{\sum_{i=1}^{N} p_{ic} g_{ic} + \epsilon}{\sum_{i=1}^{N} p_{ic} g_{ic} + \alpha \sum_{i=1}^{N} p_{i\bar{c}} g_{ic} + \beta \sum_{i=1}^{N} p_{ic} g_{i\bar{c}} + \epsilon}, \tag{3}$$

where $p_{ic}$ is the probability that pixel $i$ belongs to class $c$ and $p_{i\bar{c}}$ is the probability that pixel $i$ does not belong to class $c$. Conversely, the same is true for $g_{ic}$ and $g_{i\bar{c}}$ respectively.

Since small regions of interest (ROIs) do not contribute to the loss significantly, i.e. the value of the loss is smaller for such ROIs, the Focal Tversky Loss function (FTL) was poposed in the work of Abraham and Khan [1]. It is parametrized by $\gamma$ to switch between easy background and hard ROI training examples. The Focal Tversky Loss function is defined as:

$$FTL_c = \sum_c (1 - TI_c)^{1/\gamma}, \tag{4}$$

where $\gamma$ varies in the range $[1, 3]$. Abraham and Khan [1] hypothesize that using a higher $\alpha$ in the generalized loss function from Equation 4 improves the model convergence by shifting the focus to minimize False Negative predictions. They also mention the values they used: $\alpha = 0.7$ and $\beta = 0.3$.

3.5. **Our Semantic Segmentation Process.** The goal of our study is to solve a real problem, automatic rails identification in images, thus the data on which the chosen models will be used may come in different shapes and forms. The associated software is supposed to operate on images retrieved by a video camera placed on top of the train, aimed towards the upcoming rail track portion. It can be assumed that the format of the images is 16:9, however it is less likely that all of the feed of the cameras will be of the same resolution. Thus, we must ensure that the proposed method can be adapted to different resolutions.

In order to address this issue without changing the shape of the rails upon resizing the images to the appropriate sizes that can be fed into the network, an offline pre-processing step is performed on the semantic segmentation data.

As the original size of the images from the RailSem19 dataset is $1920 \times 1080$ and the chosen models work with squared images, the proposed crop area is a $1080 \times 1080$ square in the center of the image. An example of such a cropping can be seen in Figure 1.

This offline step allows for using images of different sizes for the inference time, which are resized to size $1080 \times 1080$ without changing the aspect ratio of the objects of interest (i.e. rails). The downside is that part of the image is left out, however the most important part containing the rails on which the train is moving on is preserved.

The images are then resized to $512 \times 512$ in order to be given as input to the models to be trained. The output will be an image of size 512x512 with black and white pixels where white pixels represent the rails and black pixels denote the background.

FIGURE 1. Example of cropping an image of size $1920 \times 1080$. The red square contains the cropped $1080 \times 1080$ area.

The ratio between background and rail annotated pixels for the cropped dataset is *36.09 : 1*, which is similar to the one obtained for the normal dataset that was *37 : 1*. Knowing this, the weight for the loss function will not be changed since the difference between 36.09 and 37 is not large.

## 4. Results

In this section we present the experiments performed during our study for the rails semantic segmentation problem in order to answer the research questions. First, we describe the metrics used for the evaluation. Afterwards we present the overfitting procedure performed to check the correctness of the selected architectures. Lastly, we describe the settings for each experiment and the obtained results.

### 4.1. **Metrics.**

In order to evaluate the obtained results, the following metrics were used:

- *Intersection over Union (IoU)* also known as the Jaccard metric [10], is the most used evaluation metric in object segmentation. It is used to determine True Positives and False Positives in a given set of predictions. True positives (TP) represent those data samples that were predicted correctly by the classifier to be a positive class, while false positives (FP) represent the samples that were incorrectly labeled as positive by the model.

Considering an input image $X$, its corresponding ground-truth mask $Y$, and the model $M_{seg}$, we can obtain $Y_p$ as $M_{seg}(\mathrm{X}) = Y_p$. In order to define the mean IoU, denoted as $mIoU$, $Y$ and $Y_p$ are used as follows:

$$mIoU = \frac{|Y \cap Y_p|}{|Y \cup Y_p|}. \tag{5}$$

This formula computes the mean IoU by considering all $K$ types of objects to be segmented. In order to compute the IoU for a single class, the following formalism can be applied. Let $k$, $k \in \{0, 1, ..., K-1\}$ be the class for which we wish to compute the IoU score, then:

$$IoU_k = \frac{|\{(i,j)|i,j \in \mathbb{N}^*, Y(i,j) = Y_p(i,j) = k\}|}{|Y \cup Y_p|}, \tag{6}$$

where the tuple $(i, j)$ can be interpreted as an $(x, y)$ coordinate in a two-dimensional image.

We have defined two metrics, one for each class: the *IoU Rail* for the rails and *IoU Background* for the background. All of the $IoU$ metrics range from 0 to 1, where 0 means no intersection and 1 means perfect overlap.

- Dice (F1) Score is based on the Dice coefficient, which was first introduced in [5]. This metric is used to measure the overlap between two samples and is equivalent to the F1 score in a binary context. The metric ranges from 0 to 1 where 1 denotes the absolute complete overlap. Using the previously defined notations, the Dice Coefficient can be expressed as in Equation 7:

$$DC = 2\frac{|Y \cap Y_p|}{|Y| + |Y_p|}. \tag{7}$$

4.2. **Experiments.** In order to check the appropriateness of the chosen architectures for the rails semantic segmentation problem we have performed different experiments. We wanted to compare the results obtained by our models with the ones available in the literature (experiments A and B) which use the RailSem19 corpus on similar data distributions. We have also tested the performance of the U-Net model on dark images and tried to enhance the performance by adding random brightness augmentations (experiment C).

The tables with results follow a similar column structure: **Model** denotes the model used for experimentation, **Parameters** denotes the number of learnable parameters corresponding to each model, **mIoU** represents the mean IoU score, **IoU Rail** and **IoU Bg** denote the IoU scores for the rails and background classes, and column **Dice** contains the Dice scores.

*Experiment A.* In the first experiment, we have compared the results obtained by our models with the ones obtained by Zendel et al. [26] in their study. Zendel et al. used a data distribution of 4000 samples, 3000 for training, 500 for validation, and 500 for testing. We did not approach the problem using this three-way split. However we have split 3500 samples selected randomly into 3000 samples for training and 500 samples for validation, and we report the results on the validation dataset. The results obtained using U-Net and ResUNet++ are showcased in Table 1.

Column **Model** contains the model trained and validated. Our models have either the w1 or w5 suffix, denoting the used weight value for the Weighted Binary Cross-Entropy loss.

TABLE 1. Rail semantic segmentation results with a 3000:500 random data distribution.

| **Model** | **Parameters** | **mIoU** | **IoU Rail** | **IoU Bg** | **Dice** |
|---|---|---|---|---|---|
| U-Net w1 | 30 M | **0.81** | 0.63 | 0.98 | 0.77 |
| U-Net w5 | 30 M | 0.80 | 0.61 | 0.98 | 0.76 |
| ResUNet++ w1 | **14** M | 0.80 | 0.61 | 0.98 | 0.76 |
| ResUNet++ w5 | **14** M | 0.78 | 0.59 | 0.98 | 0.74 |
| FRRNB [26] | 16 M | 0.71 | - | - | - |

In Table 1, it may be observed that the results obtained did manage to surpass the results obtained by Zendel et al., which are included in Table 1 in the last row. The architectures considered in our study function in a similar way to FRRNs [16] by exploiting residual connections for helping localization of pixels. Despite this, U-Nets lead to better results. Moreover, we have obtained these results without pre-training the networks on CityScapes [4]. Both U-Net and ResUNet++ surpass Zendel et al.'s results, with ResUNet++ having the least number of parameters. The U-Net model obtains slightly better results, however it utilizes approximately double the number of parameters. A size-performance trade-off must be made between the two.

For this experiment, the images size was $512 \times 512$, similar to the ones used in Zendel et al's study. The weight decay was set to $1e^{-3}$ and dropout layers were used with probability 0.2. Although counter-intuitive, the unweighted loss function with a weight of 1 leads to slightly better results in less time. An explanation for this might be that using a larger weight for the rails class, the loss is penalised harder, thus decreasing the learning speed. The models were trained for 32, 36, 22, 38 epochs respectively in the order presented in Table 1. These values were chosen after training until reaching a plateau in the loss value, meaning that no further improvements could be obtained.

The ResUNet++ architecture takes longer to reach the performance of U-Net because it has more complex components.

*Experiment B.* In the second experiment, we have compared our selected models to the RailNet architecture from Haoran Li et al. [15]. For this, 5000 images were randomly sampled for training, while the remaining 3500 were used for validation. The outcome of this experiment is given in Table 2.

TABLE 2. Rail semantic segmentation results on a 5000:3500 random data distribution.

| **Model** | **Parameters** | **mIoU** | **IoU Rail** | **IoU Bg** | **Dice** |
|---|---|---|---|---|---|
| U-Net w1 | 30 M | **0.81** | 0.64 | 0.98 | 0.78 |
| U-Net w5 | 30 M | 0.70 | 0.44 | 0.97 | 0.61 |
| ResUNet++ w1 | **14** M | 0.79 | 0.60 | 0.98 | 0.75 |
| ResUNet++ w5 | **14** M | 0.78 | 0.58 | 0.98 | 0.74 |
| RailNet [15] | 138 M | 0.54 | - | - | - |

Surprisingly, the results obtained using the 5000:3500 data distribution are considerably larger than the ones obtained by Haoran Li et al. [15]. These results were obtained using the same random distribution of samples. All models were trained with a batch size of 4 for 20 to 40 epochs. Similar to previous experiments, the models trained with a loss function weight of 1 lead to better results. The best results were selected. Resulting samples are visible in Figures 2 and 3, for U-Net and ResUNet++ respectively.

The encoder-decoder architecture with skip connections that is represented by U-Net and ResUNet++ appears to be performing better on this task than the VGG architecture combined with an Information Aggregation Module used by Haoran Li et al. [15]. It also has fewer parameters.

*Experiment C.* The last experiment was performed in order to better understand how the model behaves in dark conditions. For the training set, both daylight images and dark images were considered. To be more precise, a total of 3500 images were selected consisting of 3095 day images and 405 dark images. The train:validation split was 80:20. The training set consists of 2467 day images and 619 dark images, while the validation set contains 324 day images and 81 dark images.

Three different validation sets were considered: the one previously detailed, one with day images exclusively and one with dark images only. The results of these experiments are given in Table 3, where DN denotes day and night samples, D denotes day only samples, and N denotes night only samples.

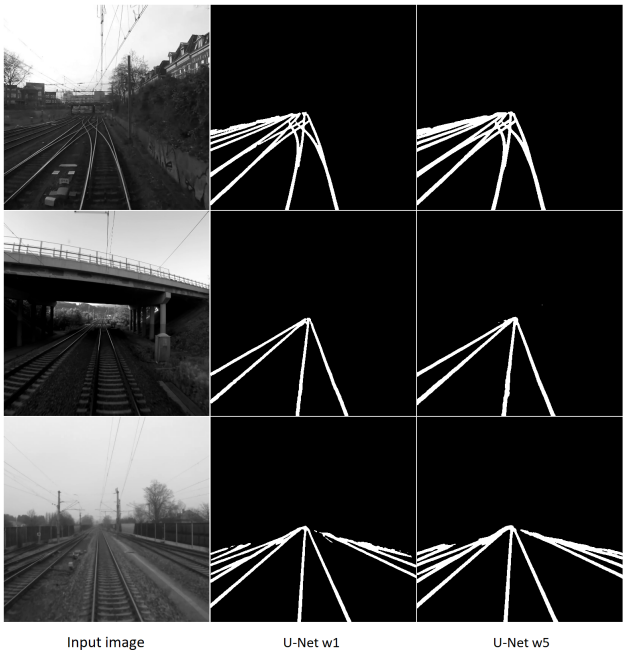



FIGURE 2. U-Net results on the 5000-3500 distribution.

TABLE 3. Results for the rails semantic segmentation task on dark images with and without augmentations.

| Val. Distribution | Model | mIoU | Rail IoU | IoU Bg | Dice |
|---|---|---|---|---|---|
| DN | U-Net | 0.77 | 0.57 | 0.97 | 0.72 |
| DN | U-Net Aug | 0.77 | 0.57 | 0.97 | 0.72 |
| D | U-Net | 0.77 | 0.57 | 0.97 | 0.73 |
| D | U-Net Aug | 0.77 | 0.57 | 0.97 | 0.73 |
| N | U-Net | 0.75 | 0.53 | 0.97 | 0.69 |
| N | U-Net Aug | 0.76 | **0.54** | 0.97 | 0.70 |

In order to address the issue of poor luminosity, one more training attempt was made with augmentations, which would randomly decrease the brightness of the grayscale images by decreasing from each pixel value a constant in order to make them darker. This constant was set to 80 after manually testing multiple values. This augmentation can be interpreted as transforming day images into night images.

As expected, the results are not as high as those for the previous experiments using approximately 3500 images in total. One of the reasons for this low

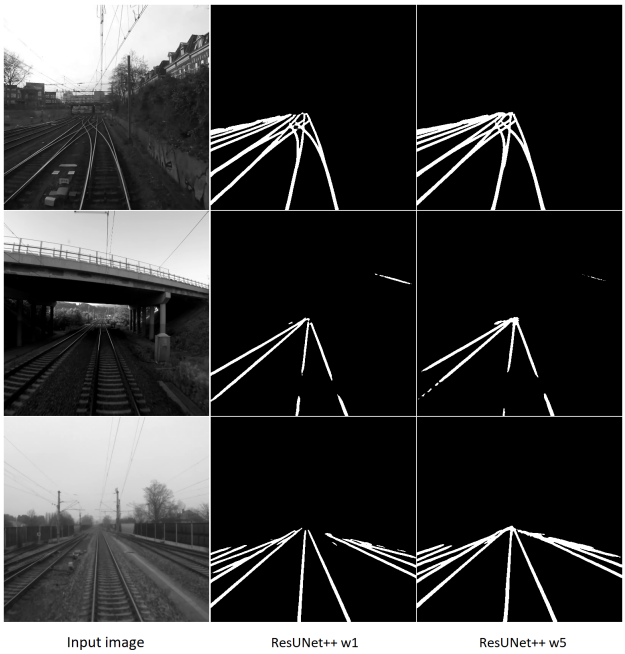



FIGURE 3. ResUNet++ results on the 5000-3500 distribution.

performance is due to the nature of the task: it is more difficult to segment rails in poor luminosity conditions.

An increase of 1.3 percentages in the Rail IoU score is observed when performing the evaluation on night-only images using random brightness augmentations. This outcome was expected since this type of augmentation helps the model learn better representations for darker images.

4.3. **Analysis.** We have observed that the U-Net architecture leads to better results than ResUNet++, although the latter uses more advanced features designed to improve its performance. Nevertheless, both architectures are appropriate for obtaining usable results for the rail semantic segmentation task. We have also observed that a smaller weight term for the Binary Cross-Entropy loss function leads to better results in less training time. This was expected since a smaller weight means a lower penalisation when the background class is predicted for a pixel instead of the rails class.

Since the variation between the results obtained using U-Net and ResUNet++ is quite small, it is difficult to proclaim a definitive winner. The decisive factor one could consider when comparing two network architectures would be the size of the network. From this perspective, ResUNet++ is deemed to

be better, having fewer trainable parameters. In the end, both architectures accomplish the task of semantic segmentation on rails well enough.

To answer the first research question, we may assume that using the ResUNet++ architecture, which has less parameters (14M) than U-Net (30M), is suitable for a dynamic environment such as the camera on the train. To add to this, the dataset used for training contains images in multiple weather conditions (snow, rain, smog) and at different times of day and night, thus increasing the usability of the trained models.

Using U-Net like architectures is the answer to the second research question, since we obtained better numerical results than three other works from the literature that aim to solve the same task. On top of the original architecture implementations, we added Dropout layers [21] and considered weighted loss functions for optimizing the models.

## 5. Conclusions and Future Considerations

In this article, an efficient solution was presented for solving the rails semantic segmentation task using deep architectures from the *U-Net* family on images taken from the perspective of the train. The considered architectures, namely *U-Net* and *ResUNet++*, led to some competitive results when compared to a selected range of related works. Our results surpassed the state-of-the-art on this task by 9 percentages.

Despite this, the task is still challenging for fine-grained semantic segmentation of rails that are further away from the camera. Other issues that are still open for research include segmenting rails in dark places (night, tunnels) or avoiding False Positives such as shadows or other objects similar to rails.

In the future, multiple aspects can be improved:

- The dataset aggregated by Zendel et al. [26] contains some images that lack proper annotations for the rails class, meaning that some rails are not annotated correctly. An improvement would be to annotate these missing rail blades and to increase the number of samples.
- From the perspective of the considered architectures, more tests can be performed with even more semantic segmentation architectures in order to compare them and select the most appropriate one for the rails semantic segmentation problem. Maybe even try a novel architecture designed especially for this task.
- Since the images of the rails are provided by a camera placed on top of the train, there is a high similarity between each frame: the rail blades are almost in the same position, but slightly shifted between consecutive frames. For this reason, an architecture that considers the previous pixels classified as rails might perform better on this task. An example of such an architecture is introduced in [25].

- It would be useful to measure the FPS of the mentioned methods and analyze what would be the most suitable hardware to be used on a real train from the perspective of power consumption and feasibility.

## References


[1] Nabila Abraham and Naimul Mefraz Khan. "A novel focal tversky loss function with improved attention u-net for lesion segmentation". In: *International Symposium on Biomedical Imaging*. IEEE. 2019, pp. 683–687.

[2] Rytis Augustauskas, Arūnas Lipnickas, and Tadas Surgailis. "Segmentation of Drilled Holes in Texture Wooden Furniture Panels Using Deep Neural Network". In: *Sensors* 21.11 (2021), p. 3633.

[3] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *TPAMI* 40.4 (2017), pp. 834–848.

[4] Marius Cordts et al. "The cityscapes dataset for semantic urban scene understanding". In: *CVPR*. 2016, pp. 3213–3223.

[5] Lee R Dice. "Measures of the amount of ecologic association between species". In: *Ecology* 26.3 (1945), pp. 297–302.

[6] Irving John Good. "Rational decisions". In: *Breakthroughs in statistics*. Springer, 1992, pp. 365–377.

[7] Christian Growitsch and Heike Wetzel. "Testing for economies of scope in European railways: an efficiency analysis". In: *Journal of Transport Economics and Policy (JTEP)* 43.1 (2009), pp. 1–24.

[8] Kaiming He et al. "Deep residual learning for image recognition". In: *CVPR*. 2016, pp. 770–778.

[9] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-excitation networks". In: *CVPR*. 2018, pp. 7132–7141.

[10] Paul Jaccard. "The distribution of the flora in the alpine zone. 1". In: *New phytologist* 11.2 (1912), pp. 37–50.

[11] Kanwal Jahan, Jeethesh Pai Umesh, and Michael Roth. "Anomaly Detection on the Rail Lines Using Semantic Segmentation and Self-supervised Learning". In: *SSCI*. IEEE. 2021, pp. 1–7.

[12] Kanwal Jahan et al. "Deep Neural Networks for Railway Switch Detection and Classification Using Onboard Camera Images". In: *SSCI*. IEEE. 2021, pp. 01–07.

[13] Debesh Jha et al. "Resunet++: An advanced architecture for medical image segmentation". In: *2019 IEEE International Symposium on Multimedia (ISM)*. IEEE. 2019, pp. 225–2255.

[14] Martin Kolařík et al. “Optimized high resolution 3D dense-U-Net network for brain and spine segmentation”. In: *Applied Sciences* 9.3 (2019), p. 404.

[15] Haoran Li et al. “RailNet: An Information Aggregation Network for Rail Track Segmentation”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–7.

[16] Tobias Pohlen et al. “Full-resolution residual networks for semantic segmentation in street scenes”. In: *CVPR*. 2017, pp. 4151–4160.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[18] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[19] L Sifre and Stéphane Mallat. “Rigid-motion scattering for texture classification”. In: *Phd. Thesis* (2014).

[20] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *International Conference on Learning Representations*. 2015.

[21] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[22] Zhen Tao et al. “Accurate and Lightweight RailNet for Real-Time Rail Line Detection”. In: *Electronics* 10.16 (2021), p. 2038.

[23] Amos Tversky. “Features of similarity.” In: *Psychological review* 84.4 (1977), p. 327.

[24] Yin Wang et al. “RailNet: A segmentation network for railroad detection”. In: *IEEE Access* 7 (2019), pp. 143772–143779.

[25] Rui Yao et al. “Video object segmentation and tracking: A survey”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 11.4 (2020), pp. 1–47.

[26] Oliver Zendel et al. “Railsem19: A dataset for semantic rail scene understanding”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 32–40.

Department of Computer-Science, *Faculty of Mathematics and Computer Science*, *Babeș-Bolyai University*, Cluj-Napoca, Romania

*Email address*: andrei.alexandrescu@stud.ubbcluj.ro

*Email address*: alexandru.manole@stud.ubbcluj.ro