Вештачка интелигенција

Artificial Inteligence

Part of Speech Tagging for Serbian language using Natural Language Toolkit

Boro Milovanović, Ranka Stanković

***Abstract*—While complex algorithms for NLP (Natural language processing) are being developed, base tasks such as tagging remain very important and still challenging. NLTK (Natural Language Toolkit) is a powerful Python library for developing programs based on NLP. We try to leverage this library to create a PoS (Part of Speech) tagger for a contemporary Serbian language. Eleven different models were created by using NLTK tagging API. The best models are transformed with the Brill tagger to improve the accuracy. We trained the models on the tagged dataset counting 180,000 tokens. The best results on the test set of 20,000 tokens were demonstrated with the Perceptron tagger: 92,52 – 95,76% accuracy for the different tagsets.**

***Index Terms*—Natural Language Processing; Machine Learning; Neural Network.**

I. Introduction

In the last couple of years, a big advancement in the field of Natural Language Processing has occurred. There are state-of-the-art language models that perform exceptionally in various language tasks [1-3]. The applications are getting broader, the algorithms are more complex [4]. Beneath the surface, there is a limited set of the tasks that still pose challenges to the researchers. Small improvements in the basic tasks pose immediate benefits to the tasks which are performed later in the pipeline.

One basic task is PoS (Part of Speech) tagging, a process of assigning a part of speech category to each token in the text. The program that performs tagging is called tagger. The taggers can be created in multiple ways. In this paper, we will create a tagger for Serbian with a help of a Python library NLTK (Natural Language Toolkit). Besides just exposing more than 50 corpora and lexical resources, NLTK is used for making programs that handle human language data, ranging from tokenization to semantic reasoning. NLTK API makes it possible to create multiple standalone tagger models as well as to combine them. We are going to try and test every model available in the version 3.5 released in March 2020. Having a plethora of different algorithms makes this library a good choice for a research.

Serbian language belongs to a group of low-resource languages so there's a modest research on this topic. First attempts to create an automatic PoS tagger for Serbian relied on a dictionary. Delić et al. used custom transformations and rules [5]. Utvić created a parameter file TT11 for a TreeTagger [6]. Later attempts relied on CRF (Conditional Random Fields) [7-8] which is among supported technologies by NLTK and will be used for training one of the taggers.

Introducing the dataset and the tagset will be done in the Section 2. The creation of multiple taggers is presented in Section 3. The results will be shown in Section 4 and briefly discussed in Section 5. We will conclude with Section 6.

II. Resources

An automated tagger is created by training on an annotated dataset. These resources are extremely valuable because they are expensive to produce. Dataset used in this paper is composed of different annotated text collections (Table I). All texts are either originally written in Serbian or translated to it. *1984* is a novel by George Orwell, part of MULTEXT-East resources [9]. *INTERA* (Integrated European language data Repository Area) is a project that produced multilingual corpus on law, health and education [10]. *Around the world in 80 days* is a novel by Jules Verne annotated during SEE-ERA.net project [11]. *ELTeC* (European Literary Text Collection) is a multilingual collection of the novels written between 1840 and 1920 [12]. *ELTeC-srp* is the Serbian part of the *ELTeC*. *History*, *Floods* and *Švejk* are three short collections originated during the same research [7]. *History* is made of several chapters from a History textbook for elementary schools. *Floods* is a newspaper collection reporting on floods in Serbia in 2014. *Švejk* is an excerpt from a novel *The Good Soldier Schweik* by Jaroslav Hašek.

TABLE I
Dataset structure

Collection	Tokens	Words
1984	108,133	69,706
INTERA	65,767	55,725
Around the world in 80 days	7,382	5,970
History	5,277	4,230
ELTeC-srp	5,118	4,236
Floods	4,671	3,813
Švejk	3,298	2,678

In total there are 199,646 tokens. Among them, 31,139 tokens are unique. An example of tagged tokens is given in the

Boro Milovanović is a PhD student attending Intelligent Systems, an interdisciplinary program at the University of Belgrade, Studentski trg 1, 11000 Belgrade, Serbia (e-mail: bmilovanovic@tesla.rcub.bg.ac.rs).

Ranka Stanković is with the Faculty of Mining and Geology, University of Belgrade, Djušina 7, 11000 Belgrade, Serbia (e-mail: ranka.stankovic@rgf.bg.ac.rs).

Table II. Every row contains 5 tab-separated values, that are described below.

TABLE II
DATASET EXAMPLE ROWS

SentenceId	Token	N_POS	UD_POS	Lemma
5	velikog	A:am	ADJ	velik
5	doba	N:n	NOUN	doba
5	.	SENT	PUNCT	.

There are 10,890 sentences in the data set, labeled by the *SentenceId*, a first value in the row. Actual word that is tagged is contained in the *Token* column. Its respective *Lemma* is in the last column. There are two PoS tags for every token, originated from two different tagsets. Tagset is a collection of tags. UD_POS is a Universal Dependency tagset [13]. N_POS is a tagset used in Serbian Morphology Dictionary [14] expanded with a gender category. From the given data we extracted token, N_POS and UD_POS tag. We stripped gender from the N_POS and got a third tagset which we called SMD_POS. All three tagsets are used in a further research.

There are 31139 unique tokens in the dataset and 17 UD PoS categories: adjective (ADJ), adposition (ADP), adverb (ADV), auxiliary verb (AUX), coordinating conjunction (CCONJ), determiner (DET), interjection (INTJ), noun (N), numeral (NUM), particle (PART), pronoun (PRON), proper noun (PROPN), punctuation (PUNCT), subordinating conjunction (SCONJ), symbol (SYM), verb (VERB) and other (X). The tags are distributed as shown in the Table III.

TABLE III
PART-OF-SPEECH CATEGORY DISTRIBUTION AND MAPPING

UD_POS	COUNT	%	N_POS
NOUN	42936	21.51%	N, N:m, N:f, N:n
PUNCT	31477	15.77%	PUNCT, SENT
VERB	20599	10.32%	V, V:m, V:f, V:n
ADJ	18949	9.49%	A, A:am, A:an, A:af
ADP	16540	8.28%	PREP
AUX	13592	6.81%	V, V:m, V:f, V:n
CCONJ	9374	4.70%	CONJ
ADV	8998	4.51%	ADV
DET	8599	4.31%	PRO, PRO:m, PRO:f, PRO:n
PART	7976	4.00%	PAR
SCONJ	6304	3.16%	CONJ
PRON	5751	2.88%	PRO, PRO:m, PRO:f, PRO:n
PROPN	3949	1.98%	N, N:m, N:f, N:n
NUM	3634	1.82%	NUM, NUM:m, NUM:f, NUM:n
X	858	0.43%	X, PREF
INTJ	110	0.06%	INT

Table III also shows mapping between UD_POS and N_POS tags. In most cases it is one-to-one relation but there are some differences between tagsets. N_POS doesn't differentiate between VERB and AUX, CCONJ and SCONJ, NOUN and PROPN. On the other hand, it separates SENT from PUNCT and PREF from X.

The most frequent word category is Noun, followed by a Punctuation and a Verb. Cumulative distribution of PoS categories is shown in Figure 1. First five categories account to 57% of all tokens. These numbers help us in creating the taggers and interpreting their performance.

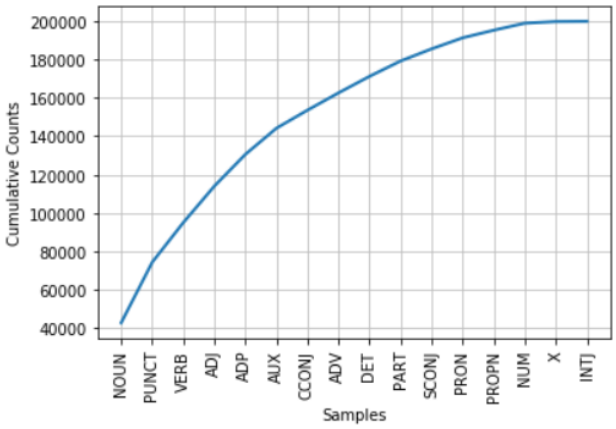


Fig. 1. Word PoS category cumulative distribution

For a complete evaluation of the taggers, we need a data from a different domain and origin. It will be exempted from the training and validation phases and will be used at the end of the evaluation phase to test generalization potential of the created models. We took this data from a *ParCoTrain* dataset [15] and mapped their tagset (which we called PCT_POS) to the SMD_POS tagset, which is shown in Table IV. This is an excerpt from a novel *Enciklopedija Mrtvih* [16], having 23,886 tokens in 946 sentences.

III. TAGGING

After the resources are ready, the process of tagging is made simple with the help of NLTK. There are a plenty of tagger models packaged in NLTK that can be trained. Every tagger has an evaluation procedure that strips down the tags from the given text, tags the text with the newly created tagger and reports the accuracy on all tokens. This measure will be used for comparing different taggers.

The simplest model in NLTK is *Default* tagger which tags every token with one selected PoS category [17]. Because the noun is by far the most represented PoS tag, the accuracy for this model on all tokens will be exactly 21,51%. This is the baseline tagger, point of reference for all other tagger models.

By applying more rules regarding the token format, a *RegExp* tagger is produced. It is initialized with the list of regex rules which are executed in the defined order. If one pattern doesn't match the given token, a next one is picked. At the end of the chain, there is a rule which states that the word is of Noun category – same as for *Default* tagger. Obviously, this model

will perform better than the baseline tagger, but making the right rules demands significant effort. We did not invest much time in producing the rules, because we had not seen significant improvements after adding new regular expressions.

TABLE IV
MAPPING BETWEEN SMD_POS AND PCT_POS TAGSETS

SMD_POS	PCT_POS
N	NOM:com, NOM: col, NOM:nam, NOM:num, NOM:approx
V	VER, VER:aux
PRO	PRO:per, PRO:intr, PRO:dem, PRO:ind, PRO:pos, PRO:rel, PRO:ref
NUM	NUM, NUM:car, NUM:ord, NUM:cal, PRO:num
A	ADJ:comp, ADJ:sup, ADJ:intr, ADJ:dem, ADJ:ind, ADJ:pos, ADJ:rel
ADV	ADV:comp, ADV:sup, ADV:intr, ADV:rel, ADV:ind
CONJ	CONJ:coor, CONJ:sub
PREP	PREP
PAR	PAR
INT	INT
SENT	SENT
PUNCT	PONC, PONC:cit
X	STR, ABR, LET, PAGE, ID

Affix tagger takes only prefixes or suffixes of fixed length in consideration. It learns the most frequent tag in the dataset for a given affix. If the word is shorter than 5 characters, tagger returns tag "None".

The most frequent PoS category is Noun but there are plenty of other words with high frequency that are not nouns. This is the idea for the *Lookup* tagger (implemented through *UnigramTagger* class). It remembers the tags for the most frequent words, while marking all the other words as None. A number of the most frequent words for which the tags should be stored is configurable. Figure 2 shows how accuracy of the model measured on the whole dataset improves with the number of the stored tags.

It is also seen that the best accuracy is achieved when the tags are remembered for all the words. This is how *Unigram* tagger behaves. It stores the most frequent PoS tag for all the tokens and marks every appearance of that token with it.

The taggers mentioned above determined the tags solely on the given token. *N-gram* tagger takes the context of the token in consideration. *Bigram* tagger takes that token and the one preceding it. *Trigram* tagger takes the two tokens before the observed one. *N-gram* taggers are most effective when combined with lower-level models.

While constructing the sequential tagger, it is possible to define a back off tagger which will take over when the current tagger is not able to determine a tag for a token (returning tag None). This is the back off chain that we tested: *Trigram – Bigram – Affix – Unigram – Default.*

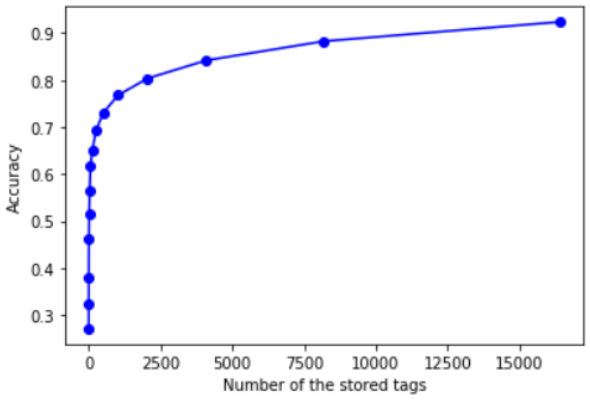


Fig. 2. Dependence of the Lookup tagger accuracy on the number of the stored tags

Aside embedded sequential tagger models available in the *nltk.tag.sequential* package, there are custom made models available in separate subpackages. *CRF* tagger is based on Conditional Random Fields [18]. *HMM* tagger is based on Hidden Markov Models [19]. Training on an averaged, one-layer neural network produces *Perceptron* tagger [20]. *TnT* tagger is short of Trigrams'n'Tags and it uses a second order Markov model to produce tags for an input sequence [21]. We declared the Unigram tagger as a back off tagger because *TnT* does not automatically work with unseen words.

Any of the mentioned taggers can be improved with the help of *Brill* tagger [22]. It uses a configurable set of rules to correct the errors and improve the total accuracy. Best performance is shown by a *brill24* set of rules. We apply *Brill* to the top performing models to try to increase the accuracy.

IV. RESULTS

All tagger models except *Default* and *RegExp* tagger are created by training on an annotated dataset. We were training on the 90% size of the original dataset size and measured accuracy on a remaining 10% size with 10-fold evaluation[1]. Results of the trained tagger models can be seen in Table V, with the accuracy calculated as

$$Accuracy = \frac{Number\ of\ correctly\ predicted\ tags}{Number\ of\ all\ tags} \quad (1)$$

The taggers are trained on an *Intel® Core™ i7-8750H CPU @ 2.20Ghz* with 16 GB RAM. It can be seen from the table that training all taggers for N_POS tagset took much more time due to the added gender information. When the processor had a task in parallel, other than training, the training times were twice as high.

Accuracy scores, for all taggers, are very close between the UD_POS and SMD_POS tagsets. This is expected because the number of the tag categories and their distribution is similar.

[1] Code for training and evaluation, example dataset and results are available at: https://github.com/bmilovanovic/pos-tagging-serbian.

However, taggers performed remarkably worse for the N_POS tagset. Information about the gender added complexity so simpler taggers could not deal with it easily. However, the best tagger models, CRF and Perceptron, kept the accuracy over 90 percent even with the gender information. We took these two taggers to the additional evaluation.

TABLE V
ACCURACY OF TAGGERS FOR EACH TAGSET

Tagger	UD_POS	SMD_POS	N_POS
Default	21.50	23.50	12.15
RegExp	23.20	25.33	13.06
Affix	88.34	87.12	81.64
Lookup	43.26	40.87	40.70
Unigram	90.56	88.79	84.87
Bigram	91.56	90.17	86.58
Trigram	91.50	90.01	86.38
CRF	**93.77**	**93.72**	**90.16**
HMM	44.28	49.80	45.58
Perceptron	**95.61**	**95.76**	**92.52**
TnT	90.83	90.51	86.95
Training Time	1143s	1343s	3074s

Useful tagger model is one which generalizes well to the text from the other domains. That's why we tested our best taggers on the text that stayed out of the training and validation phases. Results can be seen in Figure 3.

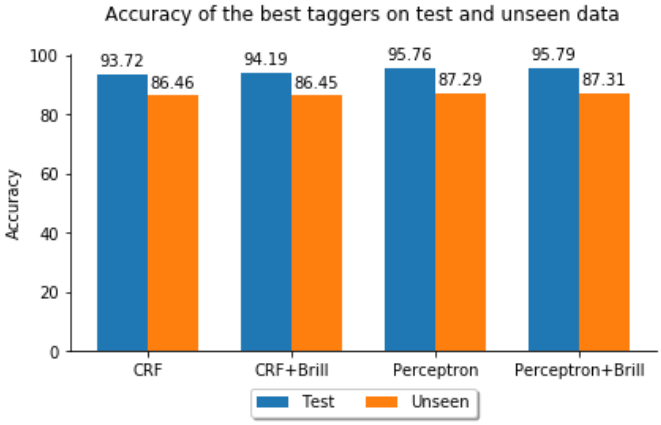


Fig. 3. Accuracy of the CRF and Perceptron variants on the test and unseen data

Taggers shown in the Figure 3 are trained on the SMD_POS tagset because it was the most like the one in unseen data so we could map between two easily. CRF and Perceptron taggers saw a small improvement with the corrections from Brill tagger. However, all four models saw a fall of about 8% in accuracy for the unseen data.

For an additional insight into the performance of the taggers, we calculate precision, recall and F1 at a tag level for the Perceptron + Brill tagger, as followed

$$Precision = \frac{T_p}{T_p + F_p} \quad (2)$$

$$Recall = \frac{T_p}{T_p + F_n} \quad (3)$$

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

The scores are calculated by iterating over all tokens, comparing the predicted tag versus actual tag and then aggregating the cases. True positive (T_p) is when these tags are identical. False positive (F_p) is the tag that is predicted but is different than actual tag. The actual tag, that is not predicted correctly, is determined as false negative (F_n). Table VI displays the results.

TABLE VI
TAG-LEVEL METRICS FOR PERCEPTRON + BRILL TAGGER EVALUATED ON UNSEEN DATA

SMD_POS	Precision	Recall	F1
N	0.91	0.99	0.92
PUNCT	0.99	0.99	0.99
V	0.91	0.93	0.92
A	0.79	0.53	0.64
PREP	0.99	0.98	0.98
CONJ	0.91	0.93	0.92
ADV	0.80	0.68	0.73
PRO	0.53	0.91	0.67
PAR	0.75	0.93	0.83
NUM	0.63	0.65	0.64
SENT	1.00	0.99	1.00
X	0.53	0.04	0.07
INTJ	0.38	0.33	0.35
Total	0.87	0.87	0.87

Many tags have low F1 scores, with X having the lowest one. Recall for X is only 0.04 which means that there are a lot of tokens with actual tag X, that the tagger did not predict as such. However, overall accuracy (87.31) is higher than most tags have because the precision is high for N and PUNCT tags which are the most frequent.

V. DISCUSSION

Looking again at the Table VI, we can notice fluctuation in performance between various tags. This is probably due to the differences in the tagging practice for the training and evaluation sets. In the process of preparing data, there are multiple tagsets and annotators. This is too many factors for an automated tagger to have the performance near maximum.

Although no research is conducted using different NLP tool and the exact same resources, there is an evidence of better

performance in PoS tagging a contemporary Serbian language [23]. Their performance on unseen data shows 0.88 precision with Spacy tagger and 0.93 with TreeTagger19 while the best tagger produced in this research achieves 0.87. The technologies in this research are not able to produce us a generalized, multi-purpose, all-around PoS tagger that can be a standard for a Serbian language.

Best performance in this research is achieved with the Perceptron tagger, a neural network which is more than a decade old. Since then, a breakthrough with deep learning has happened, so there's a strong belief that further improvements can be made with the latest neural network models [24]. However, there is a doubt if these models, because of their complexity, will ever be available in NLTK.

VI. Conclusion

We used NLTK, a Python library, to create 11 automated PoS taggers for a contemporary Serbian language. Models were trained on 180,000 tokens and evaluated on 20,000 tokens. The top performing models were improved with the help of Brill tagger and then tested on both familiar and an unfamiliar text. Best performance is shown by the Perceptron tagger: 92,52 – 95,76% accuracy for the different tagsets.

Acknowledgment

B.M. would like to thank BSc Gorana Marković for proofreading the manuscript.

References

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Oct. 2018

[2] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," Feb. 2020

[3] Z. Zhang, J. Yang, and H. Zhao, "Retrospective Reader for Machine Reading Comprehension," Jan. 2020

[4] B. Li, N. Jiang, J. Sham, H. Shi, and H. Fazal, "Real-world Conversational AI for Hotel Bookings," Proc. International Conference on Artificial Intelligence for Industries, Laguna Hills, California, USA, Sep. 2019

[5] V. Delić, M. Sečujski, and A. Kupusinac, "Transformation-based part-of-speech tagging for Serbian language," Proc. 8th WSEAS International Conference on Computational intelligence, man-machine systems and cybernetics, Tenerife, Spain, Dec. 2009

[6] M. Utvić, "Annotating the Corpus of Contemporary Serbian," INFOtheca, vol. 12 no. 2 pp 36a-47a, Dec. 2011

[7] M. Constant, C. Krstev, and D. Vitas "Lexical Analysis of Serbian with Conditional Random Fields and Large-Coverage Finite-State Resources", Proc. 7th Language and Technology Conference (LTC), Poznan, Poland, Nov. 2015

[8] N. Ljubešić, F. Klubička, Ž. Agić, and I. Jazbec, "New inflectional lexicons and training corpora for improved morphosyntactic annotation of Croatian and Serbian", Proc. 10th International Conference on Language Resources and Evaluation (LREC'16) pp. 4264-4270, Portorož, Slovenia, May 2016

[9] C. Krstev, D. Vitas, and T. Erjavec, "MorphoSyntactic Descriptions in MULTEXT-East | the Case of Serbian," Informatica, vol. 28 no. 4 pp. 431–436, Dec. 2004.

[10] M. Gavrilidou, P. Labropoulou, S. Piperidis, V. Giouli, N. Calzolari, M. Monachini, C. Soria, and K. Choukri, "Language Resources Production Models: the Case of the INTERA Multilingual Corpus and Terminology," Proc. Fifth International Conference on Language Resources and Evaluation (LREC'06), Genoa, Italy, May 2006

[11] D. l. Tufis, S. Koeva, T. Erjavec, M. Gavrilidou, and C. Krstev, (2009). "Building Language Resources and Translation Models for Machine Translation focused on South Slavic and Balkan Languages". Scientific results of the SEE-ERA.NET pilot joint call, pp 5, Oct. 2009

[12] Distant Reading for European Literary History, a COST Action funded by the Horizon 2020 Framework. https://www.distant-reading.net/, Mar. 2020

[13] M. d. Marneffe, T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, and C. D. Manning, "Universal Dependencies: A cross-linguistic typology," Proc. Ninth International Conference on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland, May 2014

[14] C. Krstev and D. Vitas, "Serbian Morphological Dictionary – SMD," University of Belgrade, HLT Group and Jerteh, Lexical resource, 2.0, 2015

[15] A. Balvet, D. Stošić, and A. Miletić, (2014). TALC-Sef a Manually-revised POS-Tagged Literary Corpus in Serbian, English and French. Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), pp. 4105-4110, Reykjavik, Iceland. May 2014

[16] D. Kiš, *Enciklopedija mrtvih*, Beograd, Jugoslavija, Globus, 1983

[17] S. Bird, E. Klein, and E. Loper, "Automatic Tagging" in *Natural Language Processing with Python*, Sebastopol, California, USA: O'Reilly, 2009, ch. 5, sec. 4, pp. 198

[18] T. Peng and M. Korobov, pythoncrfsuite. https://python-crfsuite.readthedocs.io, 2014

[19] X. Huang, A. Acero, H. Hon, "Hidden Markov Models" in Spoken Language Processing, Upper Saddle River, New Jersey: USA, ch. 8, sec. 2, pp. 378-391

[20] H. Daume III, "Frustratingly Easy Domain Adaptation," Proc. 45th Annual Meeting of the Association for Computational Linguistics, Prague, Czech Republic, June 2007

[21] T. Brant, "A Statistical Part-of-Speech Tagger," Proc. Sixth Applied Natural Language Processing Conference, Seattle, Washington, USA, 2000

[22] E. Brill, "A simple rule-based part of speech tagger", Proc. Third conference on Applied natural language processing (ANLC '92), Stroudsburg, Pennsylvania, USA, Mar. 1992.

[23] R. Stanković, B. Šandrih, C. Krstev, M. Utvić, and M. Škorić, "Machine Learning and Deep Neural Network-Based Lemmatization and Morphosyntactic Tagging for Serbian," Proc. International Conference on Language Resources and Evaluation, pp. 3954-3962, May 2020

[24] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual String Embeddings for Sequence Labeling," Proc. 27th International Conference on Computational Linguistics, Santa Fe, New Mexico, USA Aug. 2018

Visualization of Moving Objects in Thermal Image

Miloš Petrović, Nataša Vlahović and Miroslav Perić, *Member, IEEE*

***Abstract*— Motion detection in color or thermal imaging has become one of the major components of surveillance and monitoring systems. Since thermal images are usually presented as gray scale images, the need for smart assistance in surveillance for the operators has risen. A common way of emphasizing detected motion on an image is pseudo coloring. In this paper, an application for pseudo-coloring of thermal image areas with detected motion is provided in order to give an adequate visualization and draw attention of the operator to the moving objects. For motion detection SURF (Speeded Up Robust Features) detector key points are used along with Optical flow estimation (Lucas-Kanade method). Every detected region is presented by its center found by motion detection on two successive frames. Complete object motion detection and visualization is obtained using several more image processing techniques: morphological image processing, image segmentation and pseudo-coloring. Results are presented on the experimental dataset made for this purpose only.**

***Index Terms* — motion detection; thermal image; SURF; optical flow; pseudo-coloring.**

I. Introduction

Motion detection has become widely populated in military and civilian surveillance systems which has become critical for the rapid response to a certain event. Safety is one of the major problems in the 21st century, so developing this type of technology leaves many options for improvement in future use. Typical applications of motion detection are: security systems, vehicle navigation, video image reconstruction, computer vision etc. Moving object detection is the core and fundamental task of every surveillance system which is the focus of this paper.

Motion analysis is one of the most challenging tasks in digital video processing. The main challenge is to detect moving objects competently in real time. The key is to follow the change in the frames and to extract corresponding regions of the moving object and the object itself as quickly and efficiently as possible. This process becomes increasingly complicated and difficult with the presence of noise, complex backgrounds, variations in illumination, and the shadows of static and moving objects. In this paper, we propose a software image processing approach for motion detection on thermal images by retaining only the moving object of interest.

The main goal of this paper is to give an overall software image processing approach for appropriate visualization of detected motion. The rest of this paper is categorized as follows. Section 2 gives the description of the proposed object motion detection and visualization application. Section 3 shows the experimental results of our proposed system. Section 4 concludes this paper. The last section includes references.

Miloš Petrović is with the School of Electrical Engineering, University of Belgrade, 73 Bulevar kralja Aleksandra, 11020 Belgrade, Serbia (e-mail: petrovic.milos@ etf.bg.ac.rs).

Nataša Vlahović is with the Vlatacom Research and Development Institute, Bulevar Milutina Milankovića 5, 11070 Belgrade, Serbia (e-mail: natasa.vlahovic@vlatacom.com).

Miroslav Perić is with the Vlatacom Research and Development Institute, Bulevar Milutina Milankovića 5, 11070 Belgrade, Serbia (e-mail: miroslav.peric@vlatacom.com).

II. The Method

A flowchart of the algorithm for the proposed method is given in the Fig. 1. Brief description of each step of motion estimation process in the figure is discussed in the following section.

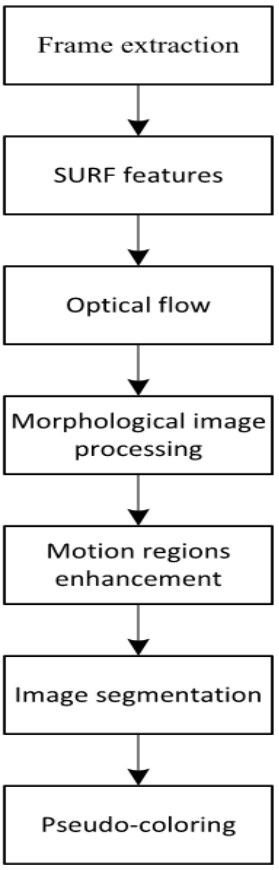


Fig 1. Flowchart of the system for moving object detection and visualization

Static thermal camera is used to capture frames that are being processed. In order to avoid occlusions, multiple different scenes are recorded. Frames are taken by Vlatacom Research and Development Institute Multi-sensor imaging system vMSIS 2 – CHD - C1200 [2]. The basic idea of Multi-sensor system is to combine data from different types of sensors (thermal sensor [3], color cameras, low light cameras [4], SWIR cameras [5], laser range finders [6], etc.). Dedicated software for image preprocessing effectively filters vibrations and rain which is the key component of external surveillance systems. This system has a pan/tilt platform and operates in enhanced temperature range (Fig. 2).

Fig. 2. Vlatacom vMSIS 2 – CHD – C1200

The SURF (Speeded Up Robust Features) algorithm [7] is used to extract the feature points from two consecutive frames. SURF is a local feature detector and descriptor. The role of the descriptors is to produce a unique description of a feature calculated from the area surrounding the point of interest. It can be used for tasks such as object recognition, image registration, classification etc. It is partly inspired by the scale-invariant feature transform (SIFT) descriptor. After SURF points extraction and matching, fixed threshold method is adopted to exclude the matching points on the targets that were not moving, the least square method is employed to solve the global motion parameters. Advantage of using SURF is reducing the number of points algorithm should be applied on which reduces computational cost compared to the algorithm applied on the whole image grid points.

Optical flow presents an apparent change of a moving object's location or deformation between frames [8,9]. Its estimation is used in many applications. Optical flow estimation yields a two-dimensional vector and motion field that represents velocities and directions of each SURF point of an image sequence. Lucas-Kanade algorithm [10] has been chosen for the estimation of Optical flow because of its high accuracy and its basic principle that uses the change of intensity between two consecutive video frames for motion detection. It assumes that the flow is essentially constant in a local neighborhood of the pixel under consideration, and solves the basic optical flow equations for all the pixels in that neighborhood by the least square criterion. Lucas-Kanade optical flow method provides visual representation of area over the moving object and the relative speed intensity of moving objects. Furthermore, points with low or zero speed vector intensity were excluded from following calculations and are considered as a part of the background which makes the algorithm more efficient.

Morphological image processing (dilation) with square structural element achieves approximate object region detection [11].

Additional enhancement of the whole process of object detection and visualization application is achieved using difference in brightness between two frames in order to match the actual object border more accurately. Precise object region detection is achieved through further morphological image processing (erosion and dilation) with square and disk structural elements of different sizes.

In combination with image segmentation with multiple thresholds [11] and pseudo-coloring (hot color map) appropriate visualization is provided. A pseudo-color image is a color image derived from a grayscale image by mapping each pixel intensity value to a color according to a table or a function. It can make some details of interest more visible.

III. Experimental Results

We implemented all of the mentioned algorithm steps (Fig 1.) in Matlab. In order to illustrate proposed algorithm one of the most suitable scenes is selected and Fig. 3 shows two consecutive frames extracted from video made by static thermal camera.

Fig. 3. Two consecutive frames recorded by vMSIS 2 – CHD – C1200

In Figure 4 SURF features are extracted and matched from two consecutive frames.

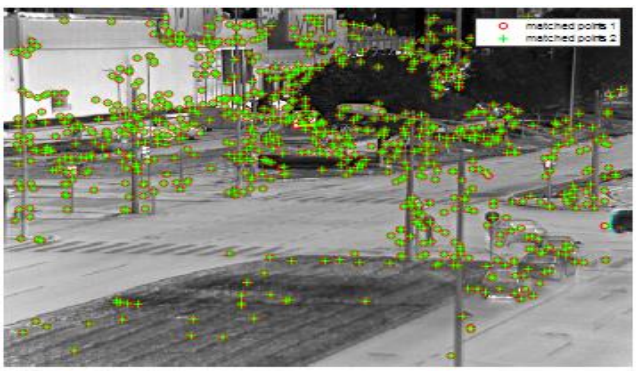


Fig. 4. Extracted and matched SURF features from two consecutive frames

Optical flow (Lucas-Kanade method) makes use of the flow vectors of moving objects over time to detect moving regions in an image which is shown in Fig. 5.

Fig. 5. Optical flow estimation based on SURF points

The following step forms a binary image where remaining SURF points, that with the highest certainty represent moving objects, were white dots on black background. The result of morphological image processing dilation with square structural element is shown in Fig. 6.

Fig. 6. Morphological image processing - dilation with square stracturale element

Aditional precision in object detection through frames brightnes difference alongside with morphological image processing – erosion followed by dilation is presented in Fig. 7.

Fig. 7. Motion region enhancement

Image segmentation is employed in order to emphasize objects of interest and separate them from the background which is illustrated in Fig. 8.

Fig. 8. Image segmentation on extracted regions

Fig. 9 shows the result of pseudo-coloring of the objects that are emphasized with the help of image segmentation.

Fig. 9. Pseudo-color image on segmented region of interest

Final result of the proposed procedure of moving object detection and visulisation is reported in the Fig. 10.

Fig 10. Motion detection and visualization – regions of interest clearly differentiated from the background

IV. Conclusion

This paper presents a design of a system for moving object detection in thermal image sequences. The proposed system employs several methods: extracting and matching SURF, estimating Optical flow by Lucas-Kanade method in reduced number of SURF points, morphological processing, segmentation and pseudo-coloring. Speeded Up Robust Features, as the most suitable to detect moving objects by the intensity changes of frames, are used as feature detectors. The combination of morphological erosion and dilation extracts significant features of region shapes from binary images after which blob analysis introduces these shapes to the next step as foregrounds (using pseudo-coloring on mentioned blobs). The detection performances are verified through the experiments based on the data using the actual footage of vehicles.

For the future work, an adaptive threshold filtering should be applied on inadequately matched SURF points and points with low velocity estimated by Optical flow, which would make this system more efficient. Also the algorithm can be developed to detect and identify overlapping objects and occlusion or transparencies during object tracking. Further research will be focused on developing a color based tracking method which can deal with both partial and complete occlusions effectively.

References

[1] Ester Martinez – Martin, Angel P. del Pobil, *"Robust Motion Detection in Real-Life Scenarios"* 1st edition, Springer, 2012.

[2] Vlatacom Research and Development Institute "Electro optical system vMSIS2-CHD-C1200", product brochure, 2016, available online, accessed on: https://1c53710f-cd70-4b75-9db4-5022cd1b5639.filesusr.com/ugd/510d2b_b84a980cf8424b3e90909639d928edd4.pdf, Accessed 2020-05-20.

[3] Helmut Budzier, Gerald Gerlach, *"Thermal Infrared Sensors: Theory, Optimization and Practice"*, Wiley, 1st edition, 2011.

[4] Takao Kuroda, *"Essential Principles of Image Sensors"*, CRC Press 1st edition, 2014.

[5] Dragana Peric, Branko Livada, *"Analysis of SWIR Imagers Application in Electro-Optical Systems"*, presented at conference IcETRAN 2017, at Kladovo, Serbia.

[6] Narain Mansharamani, *"Laser Ranging Techniques"*, BookSurge Publishing, 2018.

[7] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, *"Speeded-up robust features (SURF)"*, Comput. Vis. Image Underst., 110(3), 346-359, 2008.

[8] Burton, Andrew, Radford, John *"Thinking in Perspective: Critical Essays in the Study of Thought Processes"*. Routledge. ISBN 978-0-416-85840-2, 1987.

[9] Warren, David H.; Strelow, Edward R. "*Electronic Spatial Sensing for the Blind: Contributions from Perception*". Springer. ISBN 978-90-247-2689-9, 1985.

[10] B. D. Lucas and T. Kanade, "*An iterative image registration technique with an application to stereo vision"*, 1981.

[11] Rafael C. Gonzalez, Richard E. Woods, *Digital Image Processing*, 4th edition, Pearson/Prentice Hall, 2018.

Automatic Speech Recognition System for Dictating Medical Findings

Branislav Popović, *Member, IEEE*, Edvin Pakoci and Darko Pekar

***Abstract* — The paper presents an automatic speech recognition (ASR) system for dictating medical findings, developed by AlfaNum – Speech Technologies Ltd for the Pension and Disability Insurance Fund of the Republic of Serbia. The system is developed in a form of a distributed client-server architecture. The training of acoustic models is performed using a "chain" sub-sampled deep time-delay neural network (TDNN), while language models training is conducted using recurrent neural networks (RNNs), composed of "relu-renorm" layers followed by long short-term memory projection (LSTMP) components. The client application sends recorded user data to the server, where recognition of speech samples is performed in real time. The data is stored locally as well as in the central database, and can be exported in an appropriate form upon request. Recognition accuracy of 97% on a vocabulary of up to 50000 words is achieved.**

***Index Terms*—Automatic speech recognition, dictation, medical, Serbian, Latin.**

I. Introduction

AUTOMATIC speech recognition is a widely used technology for converting spoken words by users into text, i.e., for creating the transcription of the given conversation. Many human-machine interaction systems exist in a variety of different areas. ASR applications include dictation systems, voice assistant applications, smart homes, call centres, tools for aiding people with disabilities, and so on. As for the Serbian language, the state-of-the-art systems are constantly being upgraded. The recent research was mostly directed towards language modelling, because the previous systems had a lot of trouble dealing with the inflectivity of the Serbian language (i.e., having different cases, grammatical numbers or grammatical genders for words, which are all differentiated only by short word suffixes). The state-of-the-art Serbian language models involve deep recurrent neural networks that use embedding vectors as word representations and incorporate sub-word features, as well as additional lexical and morphological features for each word, on top of the usual "1-of-*N*" vector representation of words [1]-[2]. The state-of-the-art acoustic models, on the other hand, for several years now involve different variations of purely sequence-trained deep time-delay neural networks with subsampling, specifically designed to better model long temporal contexts [3]-[4]. These acoustic models also include accent-specific vowel models, Mel-frequency cepstral coefficients (MFCCs), pitch features and speaker identity vectors, or *i*-vectors [5], for the purpose of adaptation to different speakers and channels.

The need for training an ASR system for dictation or transcription in the medical field is not a new need – some solutions were suggested back at the end of the 20th century [6]. Most of the suggested uses include automatic speech recognition and a following transcriptionist review, and sometimes even a final physician review [7]. The main goal is to minimize clinically relevant errors, and provide reasonable general quality in practice [7]-[8]. Depending on the use (acoustic and linguistic complexity and variability, expected vocabulary size, etc.), different accuracy rates are reported, as well as different changes in department productivity. Sometimes there are changes for the worse too, as some doctors found that even though automatic speech recognition helps the overall department productivity (high gains), the length of time it takes to finish the dictation and produce the final report often increases [9]-[10]. Generally, physician surveys usually do find that most of them agree that the usage of ASR technology is a good idea, even though only a part of them (e.g. about half) report time savings [11].

The system presented in this paper is so far the only medical ASR system in existence for the Serbian language. It was created using the best available acoustic and language models mentioned above, as well as additional textual medical data obtained directly from the eventual user of the system, the Serbian Pension and Disability Insurance Fund. The system is implemented using a classic client-server architecture. The client application was specifically developed for this purpose from scratch, as it needed to fulfil very specific requirements for the Fund as well as to provide additional functionalities, like creating a final report in the specific legal form and exporting it to the central database.

The remainder of this paper is organized as follows. Section II discusses the system architecture, including the applied techniques and the used training databases. Section III is about the client application interface. Section IV describes the testing procedure and the accuracy of the system. Finally, section V gives a short recapitulation and conclusion.

Branislav Popović is with the Department for Power, Electronic and Telecommunication Engineering, Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia, Department for Music Production and Sound Design, Academy of Arts, Alfa BK University, Nemanjina 28, 11000 Belgrade, Serbia, and Computer Programming Agency Code85 Odžaci, Železnička 51, 25250 Odžaci, Serbia (phone: +381613207989; e-mail: bpopovic@uns.ac.rs).

Edvin Pakoci is with AlfaNum Speech Technologies, Bulevar vojvode Stepe 40, 21000 Novi Sad, Serbia (e-mail: edvin.pakoci@alfanum.co.rs).

Darko Pekar is with AlfaNum Speech Technologies, Bulevar vojvode Stepe 40, 21000 Novi Sad, Serbia (e:mail: darko.pekar@alfanum.co.rs).

II. System Architecture

The system is implemented in the form of a standard client-server architecture. Speech samples are sent to the ASR server to be processed and recognized. Voice activity detection is carried out implicitly, based on the most probable phoneme sequence for a given frame and the calculated signal energy. The recognition results are conveyed to the client in real time.

A. Server Side

The server recognizes chunks of audio samples. Audio content recognition is enabled for up to 15 users in parallel. Different grammars (i.e., language models) are provided, depending on the currently chosen textual field (domain of interaction), e.g., one of the fields allows dictation of medical findings in Latin, which is why a special grammar had to be trained for that specific purpose.

The baseline model is a "chain" sub-sampled time-delay deep neural network. The network is trained using cross-entropy training and a sequence-level objective function [3], [4], [12], while the training procedure consists of the pre-DNN and the DNN phase. For the pre-DNN phase, static features, including 14 Mel-frequency cepstral coefficients (MFCCs), energy and 3 pitch-related features – probability of voicing, log-pitch and delta-pitch, as well as their first and second order derivatives are extracted (the final feature vector is 54-dimensional). This phase consists of an initial flat-start monophone HMM-GMM training, triphone HMM-GMM training (targeting 3500 HMM states and 35000 Gaussians both for the first and second triphone pass), and speaker adaptive training (SAT, targeting the same model complexity). The final pre-DNN HMM-GMM model is used to provide input data alignments for the deep neural network (DNN) training. For this phase, 40 high-resolution MFCCs together with the 3 previously described pitch-based features, and a 100-dimensional speaker identity vector (*i*-vector) are used as features, producing a 143 dimensional feature vector.

The TDNN consists of 10 hidden layers, each of them containing 1024 neurons. The lower layers are trained using temporal context windows that include the preceding, the current and the following frame. The training of higher layers is conducted using also windows of 3 frames, but with 3-frame-long gaps between them. Acoustic models are trained using the recently expanded speech database for the Serbian language. The database consists of audio book recordings (recorded in a studio environment, spoken by professional speakers, 32 male and 64 female speakers, 168 hours of data), radio talk show recordings (179 hours of data, 21 male and 14 female speakers) as well as mobile phone recordings from interactions between humans and machines (requests, questions, and other inquiries, 61 hours of data, 169 male and 181 female speakers). Audio data is sampled at 16 kHz, 16 bits per sample, mono PCM. The number of speakers is increased artificially, using various combinations of speech speed and pitch modifications for similarly long chunks of data for each speaker which had enough data (398 and 420 distinct sub-speakers are obtained for audio books and radio shows, respectively, while the mobile phone speakers didn't need to be broken up). A version of the original database with a predetermined amount of added background noise was also created and incorporated into the acoustic model training. The noise recordings varied in type, from traffic and "cocktail party" noises, to construction noises, wind noises, etc [1], [12].

The language models are trained using previously anonymized real-life document examples from the Serbian Pension and Disability Fund and additional Serbian corpuses in the administrative, scientific, literary and journalistic functional styles [1], [12]. Recurrent neural network language models (RNNLMs) are used for this purpose. The network consists of 3 layers with Rectified Linear Unit (ReLU) activation functions, followed by a renormalization block (i.e., "relu-renorm"), each one containing 512 embedded neurons. LSTMP layers are injected between consecutive relu-renorm layers, while both recurrent and non-recurrent projection dimensions are set to 256. Max *n*-gram order is set to 4, therefore approximating lattice rescoring by merging histories in the lattice if they share the same 4-gram history, which prevents the lattice from exploding exponentially. The language model training is run for 30 epochs – 210 iterations based on the amount of input data. The best iteration is calculated based on the objective function value on the previously extracted validation dataset, which does not take a part in RNNLM training.

After successful initialization and authorization, the ASR server is ready to communicate with client applications. During its operation, the ASR server will print out various information in its console, such as recognized users' commands, speech detection times, confidence scores, etc. Alternatively, the ASR server can also be started as a service, without displaying the console. All the information can also be written to log files.

B. Client Side

The client interface contains several cards for the header and all the 7 standardized textual fields from the Pension and Disability Insurance Fund legal form:

1. Personal data
2. Significant allegations of the compliant
3. Medical history, physical, laboratory and other findings
4. Diagnosis (in Latin)
5. Assessment and opinion on disputable issues, as well as opinion on significant facts and circumstances not considered in the previous proceedings
6. Assessment and opinion on the correctness of findings, opinions and evaluation of expert authorities in the first instance proceedings
7. Explanation of the assessment and opinion on the correctness of the findings, opinions and evaluation of the expert authority in the first instance proceedings

Switching among the cards can be done via the appropriate keyboard shortcut, or by clicking on the desired card name (below the application toolbar). In addition to the voice input, all standard options for working with text are enabled, such as

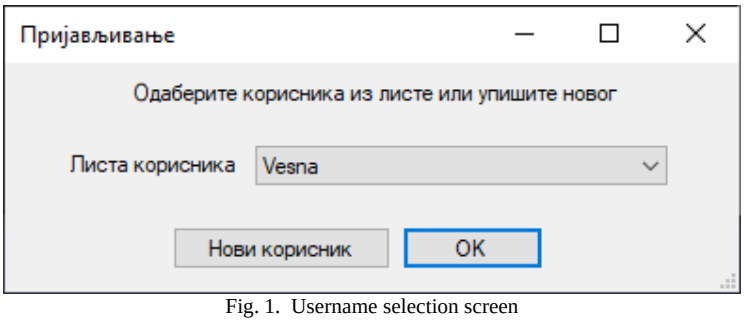


Fig. 1. Username selection screen

selecting, copying, cutting and pasting the text, changing the font size, bolding the text, as well as its conversion to the appropriate format (upper, lower, sentence or title case letters as well as letter spacing). Entries are saved automatically in the predefined folder on a local machine, as well as in the central database on a remote server. Separate subfolder is created for each of the cards, containing textual data in rich text format (RTF), together with the corresponding audio files (speech samples sent to the ASR server for recognition) and additional information about recognized words (JSON file). When saving the document, the application stores data about the medical worker (name, title, affiliation and codes) who is the current user, and creates the appropriate folder structure. This data will be linked with the username specified in the application and automatically withdrawn from the database each time a user opens the application. The interface of the client application is described in more details in the following section.

III. Application Interface

When the application is started, a username needs to be selected by choosing the appropriate name from the drop-down list presented in Fig. 1, or by entering a new username. The drop-down list is formed based on the entries previously stored in the remote database as well as the usage history of the concrete application. This name is used to identify the user communicating with the ASR server, in order to allow adaptation to the voice of a particular speaker (by linking its adaptation parameters to the selected username), therefore allowing multiple users to use the same client application with their own parameters. Speaker adaptation allows the speech recognizer to adapt the acoustic model parameters for a specific user, regardless of the gender and tone of the speaker's voice. Adaptation for any speaker should be conducted before the first recognition task. During adaptation, the user utters a predefined sequence lasting only a few seconds (a sequence of numbers in our case specifically, but other sequences would work as well). In addition to the acoustic characteristics of the voice, the signal energy level is also recorded, as well as the confidence measure for the recognized words (based on frame-level acoustic scores in the decoder). These two additional parameters are used for voice activity detection in the provided audio.

Graphical user interface of the client application is shown in Fig. 2 (personal data) and Fig. 3 (diagnosis). The recognition process on the ASR server begins by clicking on the "Start dictation" button ("Započni diktiranje" in Serbian), or by selecting the appropriate keyboard shortcut. When initiating recognition, it is important to be known which card is currently selected, so that recognition could be initiated with the appropriated grammar (i.e., language model), trained for the specific domain of interaction. If another card is clicked during recognition, the ASR server is automatically sent information to change the active language model to the one associated with the newly selected card, without having to manually stop the recognition and then restart. Recorded audio samples are sent in chunks to the ASR server, in order to enable online recognition – processing of samples begins as soon as the server accumulates enough data, and before the end of the signal, therefore allowing recognition in real time. The user ends the recognition by pressing the same button (whose text has now been changed to "End recognition" ("Kraj diktiranja")).

In addition to the final recognition result, the ASR server also allows continuous recognition – partial recognition results are provided during processing, i.e., upon voice activity detection (VAD), although these results can be modified by the decoder as new samples arrive. The final result for the previous VAD segment is determined after each long enough pause in speech (about one second or more). Both alphabets (Cyrillic and Latin) are supported, both during dictation and typing.

The client application supports a wide range of punctuation marks that are automatically converted from words during result printing, e.g., period, comma, colon, semicolon, question mark, exclamation mark, hyphen or dash, open and closed parenthesis, quotation marks, slashes, etc. – provided that the "*period*" word (i.e., "*tačka*" in Serbian) is converted only if recognized at the end of the speech segment. Automatic conversion of recognized digits, base and ordinal numbers, as well as dates is also supported. For example, the sequence "*sedmi oktobar hiljadu devetsto osamdeset prve*" (Eng. "*the seventh of October nineteen eighty-one*") will be converted into "*7.10.1981.*".

Numbers containing a decimal point can also be dictated. Furthermore, ordinal numbers (from "*first*" to "*hundredth*") followed by a slash are converted into a Roman numeral – this is used particularly in sequences such as "*po Glavi drugoj kroz B*" (Eng. "*according to Chapter second slash B*"), which will be converted into "*po Glavi II/B*" (Eng. "*according to Chapter II/B*").

Appropriate keywords can be used in combination with

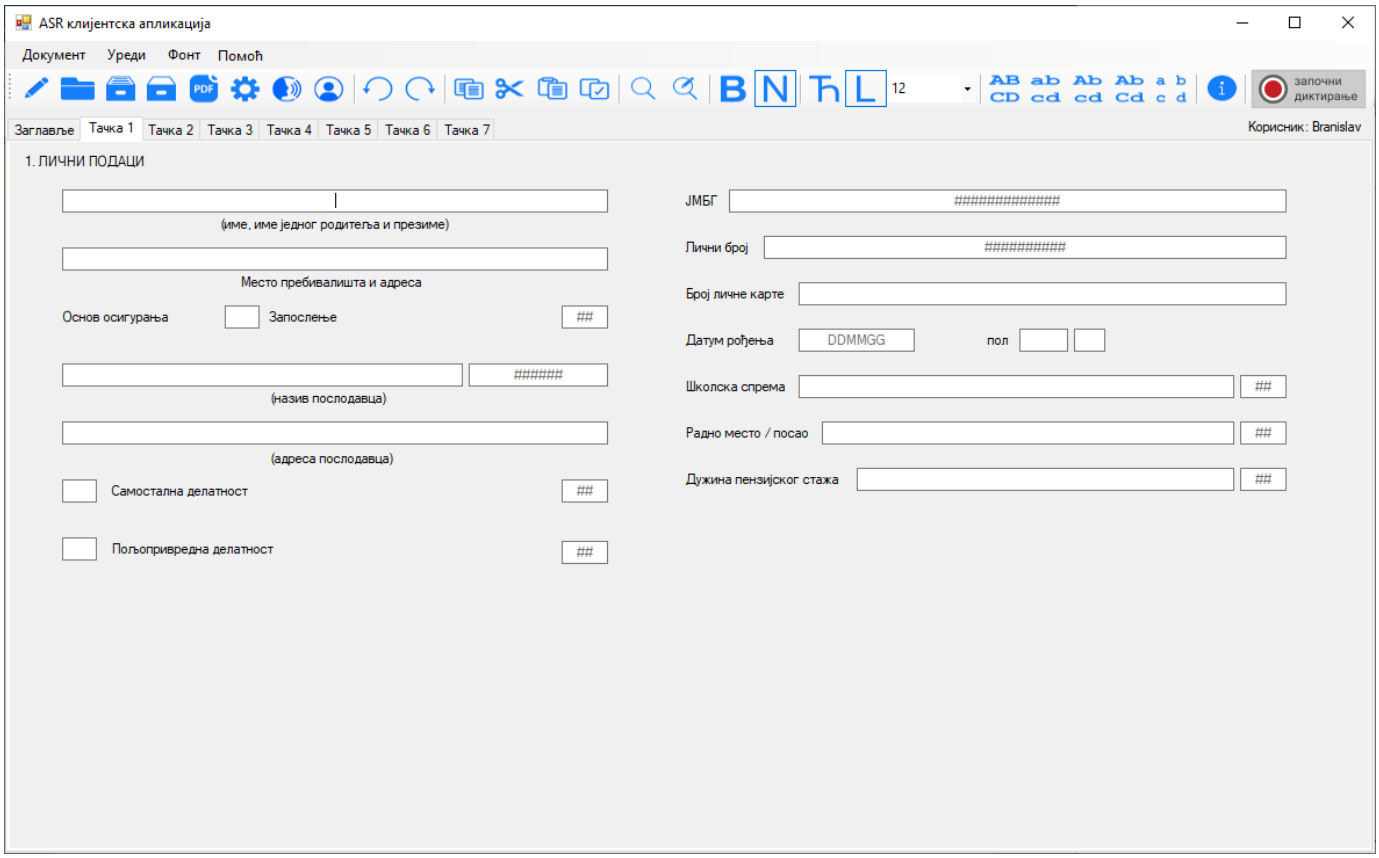


Fig. 2. Graphical user interface (personal data)

numbers. The word "*rimski*" (Eng. "*Roman*") in front of a number between 1 and 100 (either cardinal or ordinal) will make that number be written as a Roman numeral, and the command "*slovima*" (Eng. "*in text*") means that the number will be written in textual form, inside parenthesis (this needs to be done next to percentages in the Fund documents). It should also be noted that the word "*procenat*" or "*procenata*" (Eng. "*percent*") is always converted into the "%" sign, and the words "*plus*", "*minus*" and "*jednako*" (Eng. "*equals*") to the appropriate symbols "+", "-" and "=", respectively, if found next to a number. Several measurement units (meters, centimetres, millimetres, kilograms, grams, milligrams, millilitres, millimetres of mercury, per minute, per second, per litre, per square meter), are also automatically converted when recognized. Special keywords are defined for the dictation of secondary textual fields on cards 4 and 7 – "*šifra dijagnoze*" (Eng. "diagnosis code"), "*invalidnost*" (Eng. "*disability*"), "*telesno oštećenje*" (Eng. "*physical impairment*"), "*potreba za pomoći i negom*" (Eng. "*need for help and care*"), "*nesposobnost*" (Eng. "*incapacity*") and "*kontrolni pregled*" (Eng. "*control examination*"), after which one or two digits should be pronounced, or a two-digit number. A smaller set of commands such as "*obriši reč/rečenicu/paragraf*" (Eng. "*delete word/sentence/paragraph*") for correction purposes (these have to be said as a separate speech segment) and "*novi red*" (Eng. "*new line*", if said at the end of a speech segment) are also supported. There is also the "*kraj diktiranja*" (Eng. "*end dictation*") command, equivalent to clicking on the "End dictation" button – the current recording ends and all recognition results are returned to the client application.

All acronyms can be pronounced letter by letter ("*a b c d*" – Serbian Cyrillic pronunciation), or in the "singing" style, i.e. "*a be ce de*" (Serbian Latin pronunciation), and if they contain a vowel, they can be pronounced like a regular word (for example, the acronym "VOD" can be pronounced as "*v o d*", "*ve o de*", or simply "*vod*"). For some predefined acronyms and abbreviations, it is possible to pronounce whole words and still have the result written as an abbreviation, e.g. "*fundus oculi sinistri*" will be converted to "*FOS*", while "*Klinički centar*" (Eng. "*Clinical centre"*) will be converted into "*KC*".

IV. System Accuracy

The testing of the application was conducted in a relatively controlled environment (low overall and background noise), with high-quality microphones and previously prepared texts, at a time when the testers were already familiar with how to use the application. During testing, words were spoken at a normal rate, well-articulated and not overly stressed (i.e., neutral speech, no emotions in the voice). Speech speed was between 12 and 15 characters per second – neither too fast, nor too slow. Depending on the currently selected card, recognition is possible in Serbian or Latin, in real time. The recognition accuracy of about 97% on a vocabulary of about 50000 words is achieved in all domains of interaction. The accuracy was calculated in the usual way – by subtracting the word error rate (WER) from 100%, where WER is the sum of the number of word substitutions, deletions and insertions (in

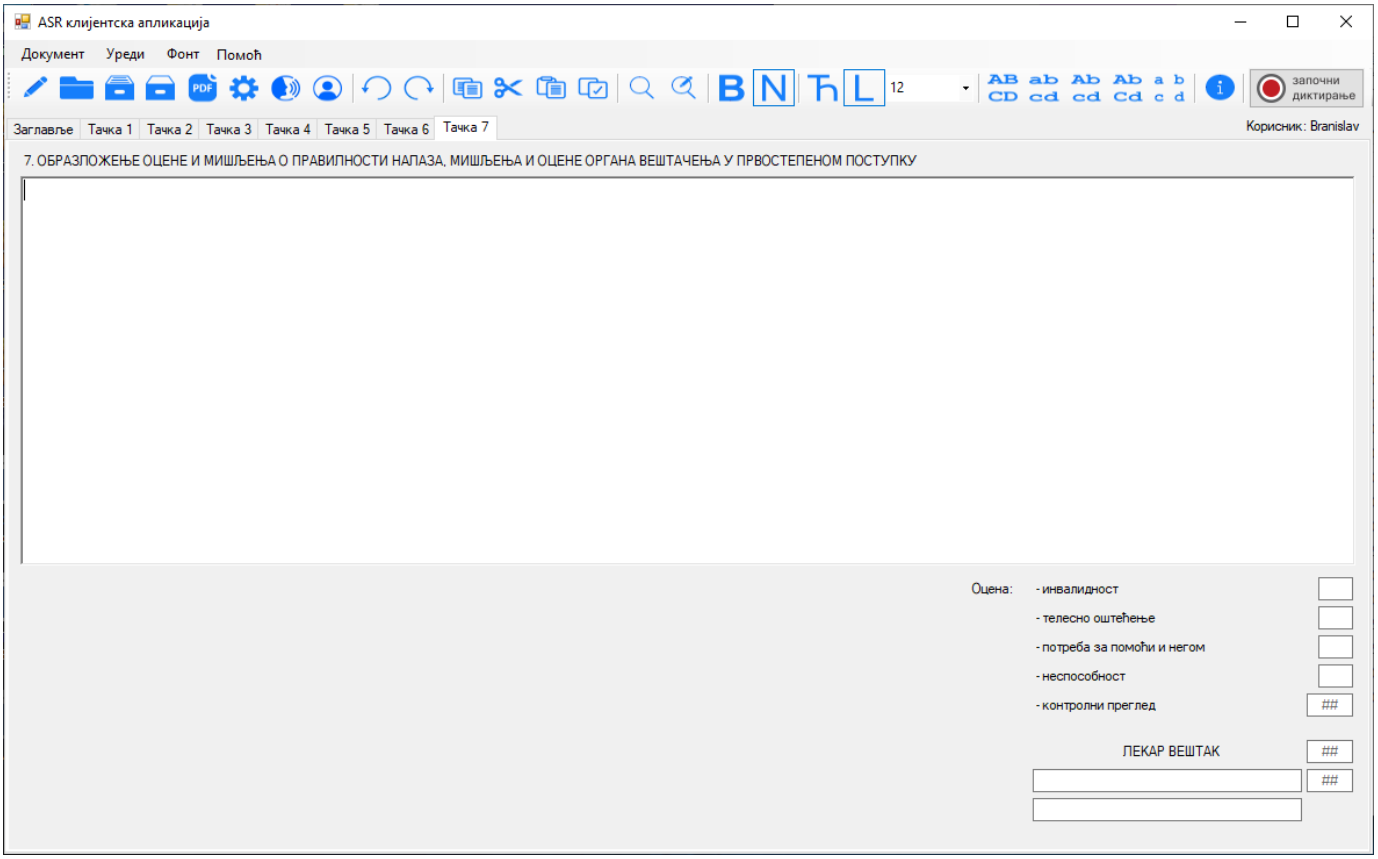


Fig. 3. Graphical user interface (diagnosis)

relation to the correct transcriptions) divided by the total number of words in the correct transcriptions. To evaluate the accuracy of the system more precisely, each digit (even in dates), punctuation mark, keyword and isolated letter were calculated as a separate word (in the same way as they are spoken). Each measurement unit is also treated as one or more separate words, depending on its transcription.

Fifteen different medical workers, both male and female, have evaluated the system accuracy, as well as its speed of returning recognized words (i.e., the real-time factor). All the speakers performed adaptation to their particular voices first. The speed of the system depends heavily on the used hardware, and the general consensus is that about 1 CPU core is needed per channel (simultaneous recognition) with a mid-range CPU (or better) to diminish the delay in obtaining the results to a manageable small value. On the other hand, the reported accuracy of 97% varied a bit from speaker to speaker, without major outliers. The typical recognition errors included out-of-vocabulary (OOV) words, as the set of possible words is naturally not a fixed or even a determinable set, so future supplementation of the language model for the purpose of covering an even larger number of words and contexts is planned. Other than that, the spelling of abbreviations produced most errors (modelling of abbreviations is a known weakness of the used acoustic model). Regardless of the few mentioned issues, the testers have rated this ASR system as a potentially very useful tool.

V. Conclusion

The system for automatic medical speech recognition in Serbian, implemented upon request of the Pension and Disability Insurance Fund of the Republic of Serbia, enables easier and faster textual input using dictation, which can simplify the work of medical workers, increase their productivity and at the same time prevent some of the typical spelling errors. The system enables detection of terms spoken in Serbian or Latin depending on the selected context, while achieving high recognition accuracy and reliable operation under specified conditions. Further improvements of accuracy can be achieved by processing a larger set of documents for the training of language models, which is a hard, highly expensive and time-consuming process, but on the other hand, significantly contributes to the robustness of recognition as well as end-user satisfaction.

Acknowledgment

The research described in this paper has been supported in part by the Serbian Ministry of Education, Science and Technological Development through the project no. 451-03-68/2020-14/200156: "Innovative Scientific and Artistic Research from the Faculty of Technical Sciences Activity Domain".

REFERENCES

[1] E. Pakoci, B. Popović and D. Pekar, "Using morphological data in language modeling for Serbian large vocabulary speech recognition," in Computational Intelligence and Neuroscience, Special Issue on Advanced Signal Processing and Adaptive Learning Methods, vol. 2019, 8 pages, 2019.

[2] B. Popović, E. Pakoci and D. Pekar, "A comparison of language model training techniques in a continuous speech recognition system for Serbian," in Proceedings of the 20th International Conference on Speech and Computer (SPECOM) – Lecture Notes in Artificial Intelligence, vol. 11096, pp. 522-531, Leipzig, Germany, September 2018.

[3] E. Pakoci, B. Popović and D. Pekar, "Improvements in Serbian speech recognition using sequence-trained deep neural networks," in SPIIRAS Proceedings, vol. 3, no. 58, pp. 53-76, 2018.

[4] V. Peddinti, D. Povey and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in Proceedings of the 16th Annual Conference of the International Speech Communication Association (INTERSPEECH), pp. 3214-3218, Dresden, Germany, September 2015.

[5] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel and P. Ouellet, "Front-end factor analysis for speaker verification," in IEEE Transactions on Audio, Speech and Language Processing, vol. 19, no. 4, pp. 788-798, 2011.

[6] D. Rosenthal, F. Chew, D. Dupuy, S. Kattapuram, W. Palmer, R. Yap and L. Levine, "Computer-based speech recognition as a replacement for medical transcription," in American Journal of Roentgenology, vol. 170, no. 1, pp. 23-25, 1998.

[7] L. Zhou, S.V. Blackley, L. Kowalski, R. Doan, W.W. Acker, A.B. Landman, E. Kontrient, D. Mack, M. Meteer, D.W. Bates and F.R. Goss, "Analysis of errors in dictated clinical documents assisted by speech recognition software and professional transcriptionists," JAMA Network Open, vol. 1, no. 3, 13 pages, 2018.

[8] C-C. Chiu, A. Tripathi, K. Chou, C. Co, N. Jaitly, D. Jaunzeikare, A. Kannan, P. Nguyen, H. Sak, A. Sankar, J. Tansuwan, N. Wan, Y. Wu and X. Zhang, "Speech recognition for medical conversations," in Proceedings of the 19th Annual Conference of the International Speech Communication Association (INTERSPEECH), pp. 2972-2976, Hyderabad, India, September 2018.

[9] I. Hammana, L. Lepanto, T.G. Poder, C. Bellemare and M-S. Ly, "Speech recognition in the radiology department: A systematic review," in Health Information Management Journal, vol. 44, no. 2, pp. 4-10, 2015.

[10] T.G. Poder, J. Fisette and V. Déry, "Speech recognition for medical dictation: Overview in Quebec and systematic review," in Journal of Medical Systems, vol. 42, no. 5, pp. 89, 2018.

[11] J.P. Lyons, S.A. Sanders, D.F. Cesene, C. Palmer, V.L. Mihalik and T. Weigel, "Speech recognition acceptance by physicians: A temporal replication of a survey of expectations and experiences," in Health Informatics Journal, vol. 22, no. 3, pp. 768-778, 2016.

[12] E. Pakoci, "Influence of morphological features on language modeling with neural networks in speech recognition systems," Ph.D. thesis, Dept. Power, Electronic and Telecommunication Engineering, University of Novi Sad, Serbia, 2019.

Classification of Chest X-Ray Images Using Deep Convolutional Neural Networks

Lara Laban, Radiša Jovanović, Mitra Vesović and Vladimir Zarić

***Abstract*—In this paper a comparison between three different types of trained VGG convolutional neural networks (CNNs) is proposed for the classification of a pediatric chest X-ray image data set. A deep convolutional neural network with an architecture resembling the VGGNet is presented using dropout, decay and data scaling. Since the dataset had a class imbalance, this was solved using a simple method called data scaling. The training of the neural network was done using small batches with a binary cross entropy loss function. The same neural network was then implemented adding batch normalization layers, and comparisons were made. Furthermore, the chest X-ray dataset was also trained using transfer learning with a pre-trained neural network VGG16 on the ImageNet dataset. Later on juxtapositions were made on using both techniques. Additionally, in applying these methods we were able to achieve a classification with the accuracy higher than 0.95 and 0.97 for the training and validation datasets, whilst incorporating only 30 epochs.**

***Index Terms*—convolutional neural networks; deep learning; transfer learning; batch normalization; chest X-ray dataset; image classification; dropout.**

I. Introduction

Convolutional neural networks (CNNs) are a subset of deep neural networks, which are used for classifying images. The main idea is to take a set of images correctly labeled as the input data and used them to train our neural network so as to achieve an output with an appropriate categorization. The inspiration for CNNs comes from the observation of the animal visual cortex. Conversely, the flourishing of these networks only came recently due to the increase of computational power and the development of many possible libraries that could be used to battle complex mathematically based problems, such as back propagation. The first paper that introduced the convolutional neural networks as we have come to know them today has [1] demonstrated that a model which consists of a multilayered network can be successfully used for recognition of stimulus patterns according to the differences in their shapes. However, there is some debate that the true begging was when a paper in 1990 [2] demonstrated that a CNN model which aggregates simpler features into progressively more complicated features can be successfully used for handwritten character recognition. In 2012 the ImageNet Large Scale Visual Recognition Challenge [3], at that moment consisting of he 1000 categories and 1.2 million images received a submission that would propel the CNNs development once again. AlexNet [4] achieved a top-5 error of 15.3% , which at the moment surpassed by an astonishing 10% all of the other submissions, and had a much faster training time as it was implemented on a GPU. The following year, the same challenge, now with a larger dataset was won by ZFNet [5]. It had the top-5 error of 14.8%, however even more so important is that it was able to reduce the first layer filter size from 11×11 to 7×7 and had a stride of 2, rather than 4 in the pooling layer.

VGG16 is a convolutional neural network model proposed in the paper [6]. This model achieved 92.7% top-5 test accuracy. The main contribution of this model was that it used 3×3 kernel sized filters, instead of the 7×7 . It was trained for weeks using GPUs, and had a huge computational cost. However, it introduced a new idea using the same kernels throughout the entire architecture, this aided in generalization for classification problems outside of what they were originally trained on. If for a second we go back to LeNet [7] that was the foundation for all of these previously mentioned CNNs we can observe the main sequence of three layers convolution, pooling and non-linearity still play the key part, and sometimes it is beneficial not to import to many layers when training a smaller dataset [8]. Finally, in recent years transfer learning [9], which addresses crossdomain learning problems by extracting useful information from data in a related domain and transferring them for being used in target tasks, has been demonstrating a significant impact.

Pneumonia is one of the main causes of death amongst children, it was stated that 19% of all deaths of kids aged 5 years and less is connected with a viral or bacterial pneumonia [10]. Today, pneumonia is the single leading cause of mortality in young children according to the World Health Organization (WHO). An even scarier report made by WHO says that 95% of new-onset childhood clinical pneumonia occurs in developing countries, many of them located in Africa and South Asia [11].

The lungs of humans are made up of small sacs called alveoli, which fill with air when a healthy person breathes, in turn when a person with pneumonia breathes the alveoli which are filled in this case with pus and fluid, blocking the

Lara Laban is with the Faculty of Mechanical Engineering, University of Belgrade, Kraljice Marije 16, 11120 Belgrade, Serbia (e-mail: llaban@mas.bg.ac.rs)

Radiša Jovanović is with the Faculty of Mechanical Engineering, University of Belgrade, Kraljice Marije 16, 11120 Belgrade, Serbia (e-mail: rjovanovic@mas.bg.ac.rs)

Mitra Vesović is with the Faculty of Mechanical Engineering, University of Belgrade, Kraljice Marije 16, 11120 Belgrade, Serbia (e-mail: mvesovic@mas.bg.ac.rs)

Vladimir Zarić is with the Faculty of Mechanical Engineering, University of Belgrade, Kraljice Marije 16, 11120 Belgrade, Serbia (e-mail: vzaric@mas.bg.ac.rs)

oxygen from arriving to the lungs, limit the capacity to intake oxygen. One of the main ways to have a proper diagnosis is radiographic data. X-rays can help in distinguishing between different types of pneumonia. However, since rapid interpretation of images is sometimes very hard, especially in developing countries, new methods are always sought after and proposed.

For that reason, an idea to battle this kind of a problem was proposed in a brilliant paper [12] published in 2018 which suggested image-based deep learning to identify various medical diagnoses, including the chest X-ray images from children. Using convolutional neural networks, transfer learning to be precise, they achieved an accuracy of 92.8% in distinguishing between normal and pneumonia images, over the course of 100 epochs. Moreover, in stating this the central goal of this paper is to try and implement a simpler CNN to combat this classification problem, all the while by using less epochs and if possible obtaining a slightly better classification accuracy.

This paper is organised in the following manner: section 2 represents a description of a dataset which is used in the training and validation of the proposed neural network. In section 3 the main methods which are used are explained in detail, as well as the architecture of the CNN. As a result, in section 4 we discuss the results and compare the methods, based on accuracy and loss functions. In section 5, following a short summary a conclusion is made and future work and possible directions are stated.

II. Dataset and its implementation

The dataset which is used in this paper consists of 5856 chest X-Ray images from children [13], including 4392 pneumonia ray (*bacterial and viral*) and 1464 normal. Being that the dataset consist of a couple of thousand pictures, there is no need to take an approach of data augmentation, where we increase the diversity of data by altering the original samples using translation, rotation, shearing, flips and adding them to the training set. However, we observe that the pneumonia part of the dataset is much larger than the part of the normal set, almost 4 times as big, resulting in a class imbalance. One way to correct this, so that our neural network may learn appropriately and not pick the pneumonia label naturally is to scale the data. This can be done by computing a weight for each class during the training, resulting in an array $[1, 3]$, and as an outcome amplifying the loss by a larger weight when we approach normal data. In this example treating an instance of normal as 3 instance of pneumonia, aids in this disproportion.

During the preprocessing of images we resized all the images to a fixed size 64×64, and in doing so we also maintained the aspect ratio. The reasoning behind this being that all the images in a dataset need to have a fixed feature vector size. This means all the images will have identical widths and heights, making it easier to quickly load and preprocess a dataset and briskly move through our convolutional neural network. The aspect ratio will enable us to resize the images along the shorter dimension, be it width or height, and in cropping it, will maintain the ratio. It is important to note that this step is not necessary if you are not working with a difficult dataset. Notwithstanding its benefits, it was implemented in this particular dataset.

A. ImageNet dataset

ImageNet is a dataset consisting of over 14 million images, which belong to one thousand classes. It was used as the dataset in the highly respected convolutional neural network model VGG16 which was proposed by Oxford scientists. In this paper the VGG16 network was used as a pre-trained convolutional neural network, in order to incorporate transfer learning and compare it to the original paper [12], mentioned beforehand, as well as the architecture that we propose.

III. Methods description

In order to try and reduce overfitting and increase our classification accuracy on the chest X-ray dataset we endeavor in performing two types of neural network training techniques:

- dropout and decay (with and without batch normalization),
- transfer learning (neural networks as feature extractors)

The first technique that is used in order to improve the generalization error in the convolutional neural network is dropout [14]. Dropout is nothing more than a form of regularization, which succors us in controlling the model capacity. The dropout layers are arranged in the network in such a manner that we have randomly disconnected nodes by a probability of 0.25 in the first few layers; and with a double increase in probability in the last layer. The reason for this is that if the first layers are dropped by a higher probability, then that will later affect the training. The dropout is implemented after the pooling layer, and before the next convolutional layer (or last flatten and dense layers). Decay that is used in this neural network is a standard decay that can be obtained using the Keras library in Python. Since the learning rate α controls the step that is made along the gradient, larger steps are usually used in the beginning to make sure that we do not stagnate in the local optima, while smaller steps are used deeper in the network and near the end of the convolution in order to converge to a global minimum. We have initialized the learning rate to be 0.05, and applied the following formula to adjust it after each epoch,

$$\alpha_{i+1} = \frac{\alpha_i}{1 + k \cdot i} \tag{1}$$

where α is the current learning rate, i is the epoch and k is the decay calculated as the division between the learning rate and the number of epochs. This type of adjustment of the learning rate each epoch, can increase accuracy, as well as reduce the loss function and the time necessary to train a network. Batch normalization [15] is used to normalize the activations of a given layer's inputs by applying mean and standard deviation before passing it onto the next layer. In addition, the covariate shift refers to a change in the

distribution of the input variables which are present in the training and validation data. Since it has been proven that the training of the neural network is the most coherent when the inputs to each layer are alike, the main intention is that even when the explicit values of inputs layers to hidden layers change, their mean and standard deviation will still remain relatively the same, thus reducing the covariate shift. Batch normalization has demonstrated an immensely effective approach to reducing the number of epochs necessary for training by allowing each layer to learn independently. Here the idea that differs from the original paper and is first proposed in [16] states that the batch normalization should be implemented after the activation layer. The main reasoning behind this is that we want to avoid setting the negative values coming out of the convolution layer to zero. Instead we pass them through the batch normalization layer, right after the activation (ReLU) layer, and assure that some of the features that otherwise would not have made it do. This yields a higher accuracy and lower loss, and is to this day a debate amongst the creators of Keras.

Finally, the second technique is transfer learning [17], a machine learning technique where networks can behave as feature extractors. Transfer learning is nothing more than the ability to use a pre-trained model to learn patterns from data, on which the original network was not trained on. As previously stated deep neural networks trained on a large scale dataset ImageNet have demonstrated to be superb at this task.

When treating networks as feature extractors we choose a point, in this case before the fully connected layer and remove it. Subsequently, in this particular example while using the VGGNet pre-trained on the ImageNet we removed the fully connected layer and stopped at the last pooling layer where the output shape is $7 \times 7 \times 512$, 512 filters with the size 7×7. Now, our feature vector has $7 \times 7 \times 512 = 25088$ values and it will be used to quantify the contents of the images, which were not included in the original training process. The format which allows us to extract these features is the hierarchical data format version 5 (hdf5), which is used to store and organize large amount of data.

Transfer learning is an optimization, which has been proven to yield a better performance and drastically save time. This is precisely why we used it in this paper, to see if we could obtain a higher classification, and perform faster. Transfer learning relaxes the hypothesis that the training data must be independent and identically distributed with the test data, which we clearly stated as a must in the beginning of this chapter. Moreover, transfer learning is able to solve the problem of insufficient training data. Furthermore, there is the option to remove the fully connected layers of the existing network in order to add a new fully connected layer to the CNN and fine tune the weights to recognize object classes. However, here it was not implemented since treating networks as arbitrary feature extractors was enough.

In the following sections we will demonstrate the architecture of a CNN that is based on VGGNet, its implementation with and without batch normalization, and additionally transfer learning will be presented instead of the CNN that was previously explained.

A. Convolutional Neural Network architecture

Into the bargain all that was explained, we picked the following CNN architecture shown in Fig. 1. It is consisted of multiple convolutional and pooling layers, as well as the fully connected layers. The first two convolutional layers learn 32 filter each with a size 3×3.

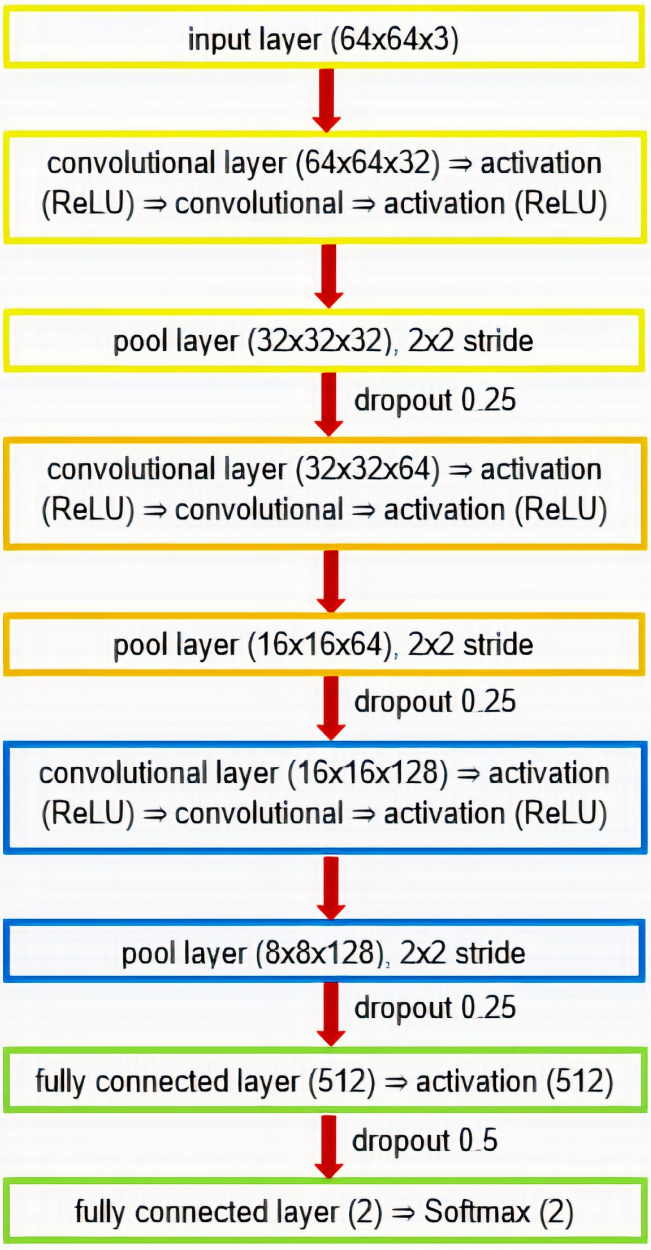


Fig. 1. A schematic of the convolutional neural network without batch normalization, that resembles the VGGNet. All of the convolutional layers that precede the fully connected layers have filters 32, 64, 128 that are the same size 3×3. The probability distribution is applied in the last layer using Softmax and the output yields two class labels normal and pneumonia.

Sequentially, the fourth and the fifth layers learn 64 filters with the size 3×3 and the last two learn 128 filters with the

size 3×3. The pool layer is used to reduce the computational load and the number of parameters, thus reducing the risk of overfitting. We used a max pooling layer with a pool size 2×2 and a stride 2. Finally, we have the fully connected layer which consists of 8192 parameters, input values which learn 512 nodes. The activation layers which were used are Rectified Linear Unit (ReLU) defined as,

$$f(x) = \max(0, x) \tag{2}$$

where x is the input into the neuron.

Softmax or the normalized exponential function assigns normalized class probabilities for each prediction, and is represented by,

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{k} e^{y_j}} \tag{3}$$

for $i = 1, ..., k$ and $\mathbf{z} = (z_1, ..., z_k) \in \mathbb{R}^k$.

Softmax takes an input vector and normalizes it into a probability distribution between $[0,1]$. Therefore the sum of all output values is equal to 1, which in turn makes the training converge more quickly. In order to achieve this, before training we must include one hot encoding in order to convert the labels from integers to vectors.

In addition, later when we want to add the batch normalization layer, we can apply it after each activation layer, as discussed previously.

B. Implementation and training of a much simpler version of the VGGNet

Taking into the bargain all that was explained before, the implementation of this CNN was done by using the Python programming language. We used Keras [18] which is mainly used for implementing of activation functions, optimizers, convolutional and pooling layers, and is actually able to do backpropagation automatically.

Right after we load and preprocess our images dataset it is necessary to use one hot encoding. This is done by using a part of the Sklearn library LabelBinarizer. However beforehand we must split the training data and the validation data, here we opted to split it 75% and 25%, sequentially. The next step is the implementation of an optimizer, here we used the stochastic gradient descent (SGD) optimizer. The SGD optimizer was set to a learning rate of $\alpha = 0.05$, with a decay in order to slowly reduce the learning rate over time and converge to the global solution more efficiently. Decaying the learning rate is beneficial in reducing overfitting and obtaining a higher classification accuracy. The smaller the learning rates are, the smaller the weight update will be enabling us to converge. The gradient descent method is an iterative optimization algorithm that operates over an optimization surface. It is a simple modification to the standard algorithm of gradient descent. The main purpose of SGD is to calculate the gradient and adjust the weights of the training data (but not on the whole dataset, but rather on a mini batch). The mini batch method is a blend of the SGD and batch methods were the neural network selects a part of the training data and updates the weights, but trains the network with the average weight update. Usually the smallest standard batch size which is used is 32, however we opted to use 24, as it complemented our data. The reasoning behind this is that present research confirms that using small batch sizes achieves the best training stability and generalization performance, for a given computational cost, across a wide range of experiments. The loss function which was used is the binary_crossentropy function. This was done because we only had two classes, if there were more we would have had to use categorical_cross_entropy, but have in mind we could have used categorical as well, but studies show that binary is much more efficient in this case.

The training was done on 30 epochs since it was enough to achieve satisfying results. After the training we implemented a method that takes the weights and the state of the optimizer and serializes them to the disc in a hdf5 format, in order to load them and test the labeling.

C. Implementation using transfer learning

The first step in this process is to extract features from VGG16, in doing so we are forward propagating the images until a given layer, and then taking those activations and treating them as feature vectors. Here the main two differences are that we used the standard a batch size of 32 and the training and test split is done at the same time as training, we again split it into 75% training data and 25% test data. Once the extraction of the features was done, we trained the classifier on those features. We also implement the GridSearchCV class to assists us to turn the parameters to the LogisticRegression classifier.

The final results are presented in the following chapter, comparisons are made and a visual representation of the graphs is shown using Matplotlib in order to estimate if there is overfitting.

IV. RESULTS AND COMPARISONS

The results of the CNN without batch normalization are presented in Table 1. We clearly see that our neural network has classification accuracy of 95%. In the following table we use the term precision which represents true positive divided by a sum of true positive and false positive, recall which represents true positive divided by a sum of true positive and false negative.

Therefore, precision is good to determine when the cost of false positives is high, on the other hand recall tells us the number of correctly labeled data. Ultimately, we have the f1-score used to find the weighted average of recall and precision. In analyzing the curves shown in Fig. 2 we see that our network learned until the 30 epoch, beyond that was simply not necessary since we already had excellent results. We can also observe that our loss and accuracy curves both almost match for training and validation, with slight deviations.

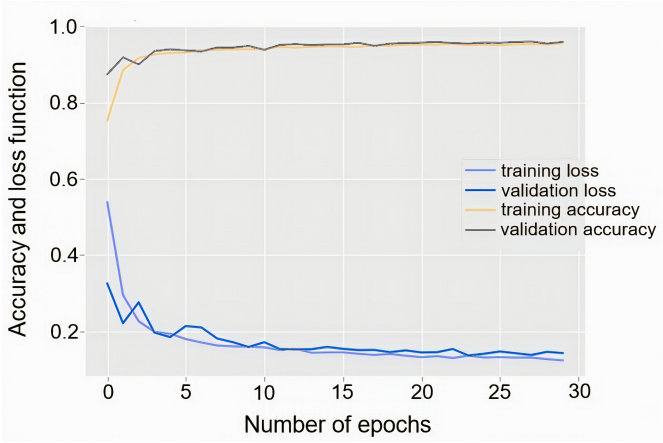


Fig. 2. A graph depicting a convolutional neural network without batch normalization, that resembles the VGGNet – training and validation loss and accuracy curves

In Fig. 3 we can see how the labeling looks, when we use the trained and saved model to label the data with this obtained accuracy.

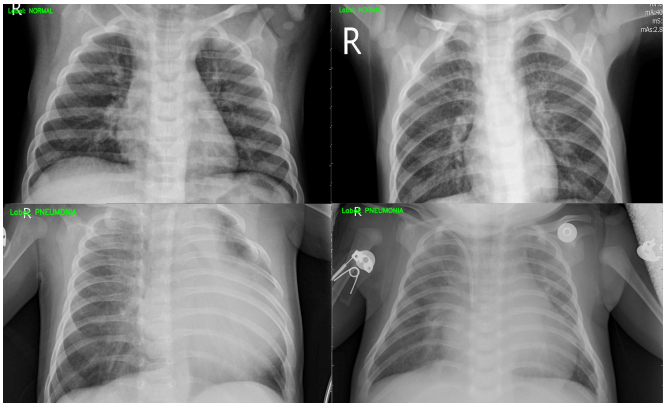


Fig. 3. The pre-trained CNN weights are loaded from the disk and make predictions for 30 randomly selected images. In the upper left and right corner we have an example of normal lungs, and in the lower left and right corner an example of pneumonia lungs.

In Table 1 we see that the CNN with batch normalization obtained the same classification accuracy of 95% after 30 epochs.

TABLE I
EXPERIMENTAL RESULTS

	precision	recall	f1-score
CNN without batch normalization			
macro avg	0.95	0.94	0.95
CNN with batch normalization			
macro avg	0.95	0.95	0.95
Transfer learning using VGG16			
macro avg	0.97	0.96	0.96

However, in analyzing the curves shown in Fig. 4 we see that our network learned until the 30 epoch, because further training past epoch 30 would result in overfitting and a wider generalization gap (loss function - the gap between the training loss and validation loss).

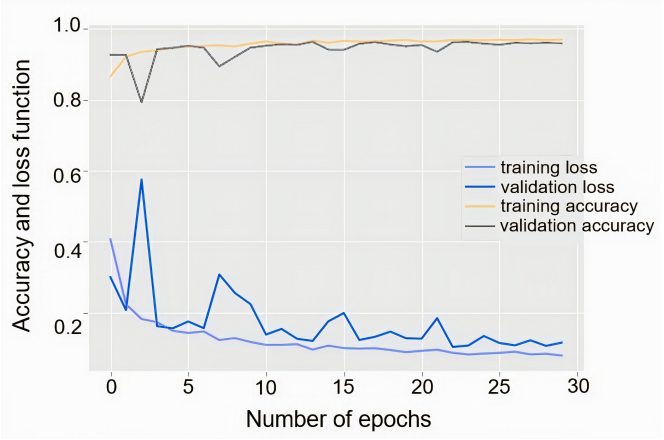


Fig. 4. A graph depicting a convolutional neural network with batch normalization, that resembles the VGGNet – training and validation loss and accuracy curves

In Table 1 we can see the results obtained by using transfer learning have a classification accuracy of 97%, which is by far the best. Furthermore, we observe that the CNN with batch normalization had a higher recall and a problem with overfitting past epoch 30, therefore the CNN without it seems like a better choice. Nevertheless, it is clear then when taking into account all three approaches we shall choose transfer learning, because not only does it yield a higher classification accuracy, but it also wasted less computational time. In addition when compared with the results of the paper [12], where transfer learning is also used and the acquired accuracy is 92.8% over the course of 100 epochs, a higher classification accuracy is obtained over the course of 30 epochs by implementing simpler CNNs and transfer learning.

V. CONCLUSION

In this paper we described three different approaches of using convolutional neural networks to classify a dataset consisting of normal and pneumonia infected lungs. We used a CNN that we constructed based on the VGGNet and implemented it with and without batch normalization. Furthermore, we used a transfer learning technique by extracting features of the neural network VGG16 trained on the ImageNet dataset. The main idea of this paper was to see if a different approach can have better results on this particular dataset, as well as see if a smaller neural network could have almost as good classification as transfer learning. The final results, when compared had a higher classification accuracy by a couple of percentages, and also achieved so in just 30 epochs, as opposed to 100 epochs, so we can conclude the goal was obtained.

Further research will focus on implementing different types of optimizers, including metaheuristic algorithms as optimizers. Also, we will focus on battling larger datasets and obtaining high classification accuracy using various methods.

ACKNOWLEDGMENT

This research was supported by the Science Fund of the Republic of Serbia, grant No. 6523109, AI- MISSION4.0, 2020-2022.

This paper was conceived within the research on the project: "*Integrated research in the field of macro, micro and nano mechanical engineering - Deep machine learning of intelligent technological systems in production engineering*", The Ministry of Education, Science and Technological Development of the Republic of Serbia (contract no. 451-03 - 68 / 2020-14 / 200105), 2020.

This work was financially supported by the Ministry of Education, Science and Technological Development of the Serbian Government, Grant TR-35029 (2018-2020).

REFERENCES

[1] K. Fukushima, S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, vol. 15, no. 6, pp. 455-469, 1982.

[2] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, L. D. Jackel, "Handwritten digit recognition with a back-propagation network," *Advances in Neural Information Processing Systems 2*, pp. 396-404, June, 1990.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Haung, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211-252, April, 2015.

[4] A. Krizhevsky, I. Sutskever, G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, no. 2, pp. 1097-1105, 2012.

[5] M. D. Zeiler, R. Fergus, "Visualizing and understanding convolutional networks," 13th European Conference, Zurich, Switzerland, pp. 818-833, September 6-12, 2014.

[6] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[7] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, n. 11, pp. 2278-2324, November, 1998.

[8] Z. Li, W. Yang, S. Peng, F. Liu, "A survey of convolutional neural networks: Analysis, applications and prospects," arXiv preprint arXiv:2004.02806, 2020.

[9] L. Shao, F. Zhu, X. Li, "Transfer learning for visual Categorization: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1019-1034, May, 2015.

[10] I. Rudan, C. Boschi-Pinto, Z. Biloglav, K. Mulholland, H. Campbell, "Epidemiology and etiology of childhood pneumonia.", *Bulletin of the World Health Organization*, vol. 86, no. 5, pp. 408–416, May, 2008.

[11] World Health Organization, Pneumonia, https://www.who.int/news-room/fact-sheets/detail/pneumonia, (last accessed 06/07/2020).

[12] D. S. Kermany, M. Goldbaum, W. Cai, C. C. S. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, J. Dong, M. K. Prasadha, J. Pei, M. Y. L. Ting, J. Zhu, C. Li, S. Hewett, J. Dong, I. Ziyar, A. Shi, R. Zhang, L. Zheng, R. Hou, W. Shi, X. Fu, Y. Duan, V. A. N. Huu, C. Wen, E. D. Zhang, C. L. Zhang, O. Li, X. Wang, M. A. Singer, X. Sun, J. Xu, A. Tafreshi, M. A. Lewis, H. Xia, K. Zhang, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122-1131, February, 2018.

[13] Public datasets; Chest X-Ray Images (Pneumonia), Version 2 https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia, (last accessed 09/03/2020).

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors", arXiv preprint arXiv:1207.0580, Jul, 2012.

[15] S. Ioffe, C.Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", Proceedings of the 32nd International Conference on Machine Learning, vol. 37, pp. 448-456, July, 2015.

[16] A. Rosebrock, *Deep Learning for computer vision with Pyhton: Starter Bundle,* 1st ed. PyImageSearch, 2017.

[17] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, C. Liu, "A survey on deep transfer learning," 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018.

[18] Keras; Python Deep Learning Library https://keras.io, (last accessed 09/03/2020).

Pneumonia Detection and Classification from X-ray Images – a Deep Learning Approach

Jelena Bozickovic, Ivan Lazic, *Member, IEEE,* and Tatjana Loncar Turukalo, *Member, IEEE*

***Abstract*— Chest X-rays are one of the first medical imaging tools used for correctly assessing different causes of pneumonia. With the recent spread of the SARS-CoV-2 virus, fast diagnostics and differentiation of the COVID-19 disease from other causes (bacterial or viral) is important. In this study we evaluated four different neural network architectures and applied transfer learning in order to try to detect and classify pneumonia in patient images in a 4-class problem (normal, viral, bacterial and COVID-19). We applied data augmentation on the outnumbered COVID-19 class and compared the effects of single end-to-end network training to a two-stage variant. The best results were obtained using the ResNet50 model with an average cross-validation accuracy of 89.97%. Across all models the COVID-19 and normal X-ray images showed very high precision and sensitivity scores.**

***Index Terms*—Chest X-ray, Deep Learning, Convolutional Neural Networks, COVID-19, SARS-CoV-2, Pneumonia**

I. Introduction

The increase and spread of registered cases of COVID-19 has accentuated the importance and accelerated development of efficient and reliable methods for diagnosis of this viral disease [1]. The fast commonly used diagnostic technique is real-time reverse transcription-polymerase chain reaction (RT-PCR) [2]. However, low sensitivity of RT-PCR test (60%–70%), and the deficit of the tests in developing countries emphasize the role of chest radiology (computed tomography and X-ray) and blood analysis in diagnostics and timely treatment [3]. The main features of pneumonia caused by SARS-CoV-2 on the X-ray images are peripheral and lower lobe predominant rounded airspace opacities and multifocal rounded opacities and nodules [4] with multifocal non-peripheral airspace opacities being less pronounced as a feature.

The urgency created by the COVID-19 pandemic requires fast diagnostics and differentiation of the COVID-19 caused pneumonia cases from other such as bacterial or viral). Early detection of pneumonia development is another goal, as it largely influences the flow and consequences of the disease. Even before the COVID-19 cases, pneumonia was one of the leading causes of death among children under 5 years old and lower respiratory tract infections (LRTI) responsible for 2.8 million deaths annually [5]. As one of the first examinations when COVID-19 or LRTI are suspected, both X-ray or CT scans provide useful resources for a machine learning approach to pneumonia detection and classification. The abundance of new cases facilitate data gathering purposes and support further improvements towards development of an automatic diagnostic support model.

Deep learning techniques have already been applied to the problem of pneumonia detection in X-ray images [6] showing that an accurate deep learning model can assist in diagnosis, especially when medical expertise or experience are insufficient. COVID-19 has been an incentive to speed up the progress in this area. The literature review from 2020 exhibits the performance of many well-known pre-trained architectures in pneumonia detection and classification tasks. ResNet50, InceptionV3 and InceptionResNetV2 pre-trained models with ImageNet [7] data were used in [8] with an average accuracy of 98%, 97% and 87% for the three models respectively for a binary classification problem between normal and COVID-19 cases with only 50 samples per class. In [9] the pretrained 121-DenseNet model was used, as in [6], achieving an accuracy of 87.2% for a 4-class classification problem between normal chest X-ray scans and 3 different pneumonia cases caused by bacteria, viruses or COVID-19. However, the dataset used in [9] was heavily imbalanced with 155 sample images of the COVID-19 class whereas the images for other classes were taken from [10]. Regarding the same 4-class classification problem with a balanced dataset of around 300 samples per class, authors in [11] used the Xception model pre-trained on the ImageNet data, where they achieved an accuracy of 89.5%. In these works, a small set of COVID-19 image data was available, both [8] and [11] focused mainly on end-to-end training of the used pre-trained models without resorting to data augmentation techniques.

In this work we evaluate several ImageNet pretrained models on the 4-class classification task using an expanded chest X-ray dataset from healthy patients and patients with bacterial, viral and COVID-19 induced pneumonia. Additionally, we compare the results obtained from end-to-end training of the models and a two-stage training process associated with transfer learning and fine-tuning.

Jelena Božičković – Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovića 6, 21102, Novi Sad, Serbia (e-mail: jelena.bozickovic@live.com).

Ivan Lazić – Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovića 6, 21102, Novi Sad, Serbia (e-mail: ivan.lazic@uns.ac.rs).

Tatjana Lončar Turukalo - Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovića 6, 21102, Novi Sad, Serbia (e-mail: turukalo@uns.ac.rs).

II. Materials and Methods

A. Database

In this study two publicly available databases were used. Images representing classes of normal X-rays, viral X-rays and bacterial X-rays were collected from the Chest X-Ray Images (Pneumonia) database [10], while the COVID-19 images class were collected from the COVID-19 Radiography Database [12].

The databases contain a total of 5669 samples, of which: 1575 samples from the class of normal X-rays, 2530 samples from the class of bacterial images and 1345 from the class of viral images. The COVID-19 class, however, only had 219 samples which were taken from 137 patients. The initial database used in this research was designed so that 15% of the patients with COVID-19 were moved to the validation and test set with only 1 unique image per patient. The same number of X-ray images are taken from other classes for the validation and test set as well. The rest of the image samples were used as the training set. The number of samples by class is shown in Table I. As the number of samples of the COVID-19 disease class was significantly smaller than the number of samples in other classes, the COVID-19 samples were augmented using random rotation by 15º, zooming in the range from 0.8 to 1.2 pixels, image rotation around the vertical axis and translation by up to 0.1 fraction of total width and height of the image. This resulted in a database with approximately balanced classes. All of the input images were scaled to 224x224 pixels. A sample image for each class is presented in Fig. 1.

B. Baseline deep learning models

Deep learning is a powerful framework for supervised learning which benefits from adding more layers and more units to achieve excellent performance in modeling complex functions, given sufficiently large labeled dataset [13]. Large datasets facilitate the use of larger models, offer better generalization, with the burden placed on the training process in terms of time and computational power, which is balanced by advances in hardware, software and parallelization [14].
For smaller data sets, overfitting can be prevented using pre-trained network models [15], which are usually trained on very large datasets, such as the ImageNet database, and used for feature extraction.

In this study, 4 different pretrained models were evaluated: ResNet50 [16], InceptionV3 [17], InceptionResNetV2 [18] and Xception [19].

TABLE I
NUMBER OF SAMPLES PER CLASS FOR THE INITIAL DATABASE

Set / Class	Training	Test	Valid.
Bacterial	1308	21	21
COVID19 augmented	1304	21	21
Normal	1310	21	21
Viral	1290	21	21

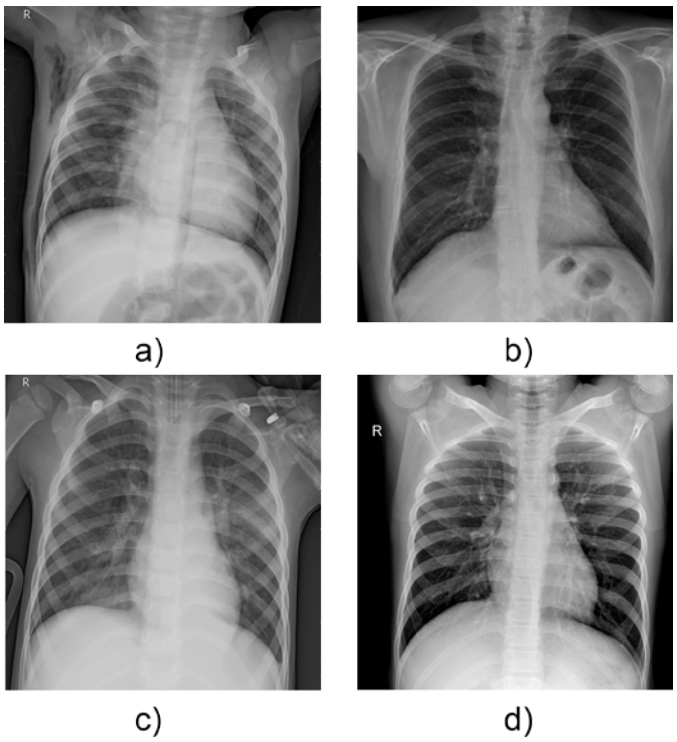


Fig. 1. Sample input from a) bacterial, b) COVID-19, c) viral and d) normal X-ray images.

The ResNet50 [16] model is based on a residual training mode to simplify the learning of deep neural networks. The network architecture involves reformulating the layers so that they learn the residual functions depending on the input layers. The depth of the residual network is 8 times larger than the VGG [20] network, but it is less complex.

The InceptionV3 [17] model allows for an expansion of depth and width of deep neural networks in a way that does not require more computing power. The model generates features on several levels using 1x1, 3x3 and 5x5 convolution filters.

InceptionResNetV2 [18] is a model that combines Inception models and residual models. It has been shown that training with residual connections significantly speeds up training compared to the Inception model itself. It has also been proven that the combination of these two models gives better results compared to the individual models.

The Xception [19] model represents such an architecture of a convolutional neural network in which the convolutional layers are completely separated. Specifically, the hypothesis behind the Xception model architecture is as follows: mapping correlations between channels and spatial correlations in feature maps can be completely separated. Network architecture consists of linearly arranged separable convolutional layers with residual connections.

All of the models show exceptional results on the ImageNet dataset classification problem, making them powerful feature extractors and classifiers. Using the stored model weights as the learned knowledge, the networks can be applied on a new set of data using transfer learning by detaching the original classification layers and training only the specific classification layers needed for the required 4-class problem.

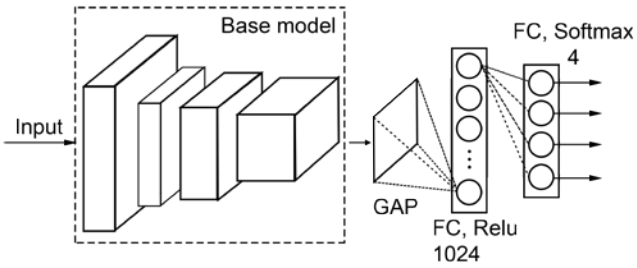


Fig. 2. Deep network architecture used for the experiments.

The models' performance can then be additionally improved by fine-tuning the original baseline model weights to the input data, as the original set of classes didn't include X-ray images. This allows the network to adjust its original weights to the new data, thus enabling it to better suit the new classification problem. This is usually done with a lower learning rate to prevent the model from completely forgetting the valuable knowledge initially learned on the large dataset.

C. Network architecture and model training

The network architecture used in this work is presented in Fig 2. After one of the mentioned base models, a classification layer is added which consists of a global average pooling (GAP) and two fully connected (FC) layers. The models were constructed using the Tensorflow 2.0 library.

Three different strategies were used to train the classifier with each baseline model:

- the first approach involved unlocking all layers of the pre-trained model.
- the second involved a two-step training procedure: in the first step, the layers of the base model were frozen, while in the second step, these layers were fully trainable.
- the third additionally introduced a dropout layer before the classification layers and performed two-step training as in the second experiment.

All models were trained using all three strategies. In the first experiment for all baseline models, the parameters for the Adam [21] optimizer, the learning speed and the epsilon parameter were set to 10^{-5} and 0.1, determined empirically. For the two-step training methods, the initial training of classification layers was done with a default learning rate of 0.001, while the second training was done with a learning rate of 10^{-5} and epsilon of 0.1. The size of the batch was 32. In the training process the early stopping method was used, which monitored the validation loss and stopped the training if the validation loss didn't improve for 15 epochs. With the obtained optimal epoch number, new models and datasets were constructed from the original images in a cross-validation fashion. The number of folds was 5 and the split between the new training and test sets was 80%-20% with data augmentation being applied to the COVID-19 class in a similar way as with the initial training. In this case, there wasn't a need for a validation set as model parameters were already determined in advance.

TABLE II
AVERAGE MODEL EVALUATION METRICS

		Experiment 1				Experiment 2				Experiment 3			
Model	Class	Accuracy [%]	Precision [%]	Sensitivity [%]	F1	Accuracy [%]	Precision [%]	Sensitivity [%]	F1	Accuracy [%]	Precision [%]	Sensitivity [%]	F1
ResNet50	0	87.75	71.53	83.49	0.77	87.41	74.37	82.62	0.78	89.97	80.26	84.90	0.82
	1		98.52	97.26	0.98		97.04	98.62	0.98		97.78	99.31	0.99
	2		96.32	97.22	0.97		96.38	93.07	0.95		99.26	95.19	0.97
	3		84.63	75.31	0.79		81.85	77.12	0.79		82.57	81.42	0.82
InceptionV3	0	87.96	75.21	81.92	0.78	86.07	72.46	82.37	0.77	87.43	76.64	83.77	0.80
	1		96.30	97.88	0.97		96.32	97.83	0.97		96.32	98.60	0.97
	2		96.32	93.17	0.95		94.89	91.98	0.93		95.61	91.81	0.94
	3		84.02	79.49	0.81		80.32	74.21	0.77		81.14	77.15	0.79
Inception ResNetV2	0	87.41	73.02	85.82	0.79	85.40	73.76	77.52	0.76	87.96	74.42	85.49	0.79
	1		95.61	97.83	0.97		95.58	98.51	0.97		95.58	99.31	0.97
	2		95.58	93.74	0.95		94.07	92.91	0.93		94.84	94.23	0.94
	3		85.42	74.59	0.80		78.17	73.84	0.76		87.01	76.32	0.81
Xception	0	89.48	78.23	87.77	0.82	87.86	76.80	86.42	0.81	87.51	73.76	87.13	0.80
	1		97.78	97.11	0.97		94.84	98.60	0.97		96.30	98.62	0.97
	2		95.66	93.91	0.95		97.12	91.17	0.94		98.52	89.96	0.94
	3		86.27	80.80	0.83		82.67	78.03	0.80		81.77	76.19	0.79

A new model is then trained for each variant of the baseline network and for each of the three strategies and for each testing fold of the cross-validation.

III. Results

Table II presents the average evaluation results for different architectures and training strategies over different folds using accuracy, precision, sensitivity and F1 measure. The classes numbers from 0 to 3, correspond to bacterial, COVID-19, normal and viral class respectively. In the case of the ResNet50 and InceptionResNetV2 models, the performance of the models indicated the improved performance when the two-step training strategy was used. The pre-trained ResNet50 model has achieved the overall best results with an accuracy of 89,97%, whereas the InceptionResNetV2 had an accuracy of 87,96%. In most cases, adding the dropout layer before the classifier improved the two-step training process. The Xception and InceptionV3 models achieved their best score of 89,48% and 87,96% using the first training strategy. Overall, the best precision and sensitivity is obtained on the COVID-19 class, with the normal class following it. The most difficult task proved to be distinguishing between the bacterial and viral classes.

IV. Conclusion

From an engineering point of view, the research showed that it is possible to use pre-trained architectures in order to detect and classify different types of pneumonia, including COVID-19. In general, the two-stage training strategy provided better results in almost all of the baseline models. The analysis of confusion matrices indicated COVID-19 X-ray scans can successfully be differentiated from images of viral pneumonia. As the network is trained and tested on a limited input of COVID-19 cases, improvements are expected by including more patient images to the current database. Additionally, further testing can be done on other non-explored pretrained deep neural models such as the EfficientNet series, NASNet or DenseNet models, as well as trying out simpler models.

Acknowledgment

This research has been supported by the Ministry of Education, Science and Technological Development through the project no. 451-03-68/2020-14/200156: “Innovative scientific and artistic research from the FTS domain” and COST Actions CA15120 and CA19136.

References

[1] “Coronavirus Disease (COVID-19) – events as they happen”, www.who.int, 2020. [Online]. Available: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/events-as-they-happen

[2] Z.Y. Zu, M.D. Jiang, P.P. Xu, W. Chen, Q.Q. Ni, G.M. Lu, L.J. Zhang, “Coronavirus Disease 2019 (COVID-19): A Perspective from China,” *Radiology*, vol. 296, no. 2, pp. E15–E25, Feb. 2020, doi: 10.1148/radiol.2020200490.

[3] J. P. Kanne, B. P. Little, J. H. Chung, B. M. Elicker, and L. H. Ketai, “Essentials for Radiologists on COVID-19: An Update—Radiology Scientific Expert Panel,” *Radiology*, vol. 296, no. 2, pp. E113–E114, Feb. 2020, doi: 10.1148/radiol.2020200527.

[4] UCLA Radiology, “COVID-19 Chest X-Ray Guideline”, Los Angeles, Westwood, Manhattan Beach, Santa Monica, CA. Available: https://www.uclahealth.org/radiology/covid-19-chest-x-ray-guideline.

[5] GBD 2015 Mortality and Causes of Death Collaborators, “Global, regional, and national life expectancy, all-cause mortality, and cause-specific mortality for 249 causes of death, 1980–2015: a systematic analysis for the Global Burden of Disease Study 2015”, Lancet 2016; 388: 1459–1544

[6] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, M. P. Lungren, A. Y. Ng, “CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning,” *arXiv:1711.05225 [cs, stat]*, Dec. 2017, [Online]. Available: http://arxiv.org/abs/1711.05225.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.

[8] A. Narin, C. Kaya, and Z. Pamuk, “Automatic Detection of Coronavirus Disease (COVID-19) Using X-ray Images and Deep Convolutional Neural Networks,” *arXiv:2003.10849 [cs, eess]*, Jul. 2020, [Online]. Available: http://arxiv.org/abs/2003.10849.

[9] A. Mangal, S. Kalia, H Rajgopal, K. Rangarajan, V. Namboodiri, S. Banerjee, C. Arora, “CovidAID: COVID-19 Detection Using Chest X-Ray,” *arXiv:2004.09803 [cs, eess]*, Apr. 2020, [Online]. Available: http://arxiv.org/abs/2004.09803.

[10] D. S. Kermany, K. Zhang, and M. H. Goldbaum, “Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification,” 2018, doi: 10.17632/RSCBJBR9SJ.2.

[11] A. I. Khan, J. L. Shah, and M. M. Bhat, “CoroNet: A deep neural network for detection and diagnosis of COVID-19 from chest x-ray images,” *Computer Methods and Programs in Biomedicine*, vol. 196, p. 105581, Nov. 2020, doi: 10.1016/j.cmpb.2020.105581.

[12] M. E. H. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. Abdul Kadir, Z. Bin Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. Al-Emadi, M. Bin I. Reaz, T. I. Islam, “Can AI help in screening Viral and COVID-19 pneumonia?,” *arXiv:2003.13145 [cs]*, Jun. 2020, [Online]. Available: http://arxiv.org/abs/2003.13145.

[13] I. Goodfellow, Y. Bengio, and A. Courville, “Deep learning”, MIT Press, Nov. 2016.

[14] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436-444, 2015. Available: 10.1038/nature14539

[15] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013, doi: 10.1109/TPAMI.2013.50.

[16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *arXiv:1512.03385 [cs]*, Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.03385.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *arXiv:1512.00567 [cs]*, Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.00567.

[18] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning,” *arXiv:1602.07261 [cs]*, Aug. 2016, Available: http://arxiv.org/abs/1602.07261.

[19] F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” *arXiv:1610.02357 [cs]*, Apr. 2017, Available: http://arxiv.org/abs/1610.02357.

[20] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv:1409.1556 [cs]*, Apr. 2015, [Online]. Available: http://arxiv.org/abs/1409.1556.

[21] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *arXiv:1412.6980 [cs]*, Jan. 2017, Available: http://arxiv.org/abs/1412.6980.

Identifikacija vrsta za potrebe biomonitoringa korišćenjem konvolucionih neuronskih mreža i dubokog učenja

Aleksandar Milosavljević, Đurađ Milošević i Bratislav Predić

***Apstrakt*—Akvatični insekti i drugi bentonski makrobeskičmenjaci uglavnom se koriste kao bioindikatori ekološkog stanja slatkih voda. Međutim, skup i dugotrajan postupak identifikacije vrsta predstavlja jednu od ključnih prepreka za pouzdan biomonitoring akvatičnih ekosistema. U radu je predložen metod za identifikaciju vrsta zasnovan na dubokom učenju čija je evaluacija obavljena na nekoliko javno dostupnih skupova podatak (FIN-Benthic, STONEFLY9 i EPT29) kao i na sopstvenom CHIRO10 skupu podataka. Predloženi metod se zasniva na tri tehnike dubokog učenja koje se koriste za poboljšanje robusnosti kada se za obučavanje koristi relativno mali skup podataka: preneseno učenje (eng. transfer learning), proširivanje podataka (eng. data augmentation), kao i odbacivanje (eng. dropout). Evaluacija modela je vršena korišćenjem ulaznih slika dimenzija 256x256 piksela gde je 50% slika korišćeno za treniranje, 20% za validaciju, a 30% za testiranje. Dobijeni rezultati pokazuju značajno poboljšanje u odnosu na tradicionalne metode koje su originalno korišćenje i potvrđuju da postoji značajan dobitak kada postoji veći broj slika po uzorku.**

***Ključne reči*—Duboko učenje; konvolucione neuronske mreže; klasifikacija slika; preneseno učenje; proširivanje podataka; biomonitoring; akvatični insekti.**

I. Uvod

Raznolikost gena, vrsta i ekosistema opada globalno brže nego ikad pre u ljudskoj istoriji [1]. Akvatični ekosistemi prikazuju među najvećim stopama opadanja sa alarmantnim gubitkom biodiverziteta, te je potreba za isplativim alatima za biomonitoring time i veća.

Tradicionalni pristup morfološkoj identifikaciji u biomonitoringu pretpostavlja korišćenje što šire taksonomske rezolucije [2]. Međutim, identifikacija makrobeskičmenjaka na osnovu morfoloških karakteristika može biti problematična jer broj pogrešno klasifikovanih vrsta raste sa povećanjem taksonomske rezolucije. S druge strane, morfološka identifikacija je vremenski zahtevan i skup proces, te je kao takav neprimenjiv za rutinski biomonitoring [3], [4]. Pobrojani nedostaci tradicionalnog biomonitoringa stvaraju potrebu za razvojem alternativnih pristupa za obradu uzoraka. Nedavni napredak u računarskom vidu u pogledu klasifikacije slika korišćenjem konvolucionih neuronskih mreža (KNM) i dubokog učenja otvorili su put pouzdanoj automatizaciji procesa identifikacije.

Klasifikacija slika u računarskom vidu je problem gde se na osnovu skupa slika određenih kategorija izgrađuje model sposoban da predvidi kategoriju, sa određenom tačnošću, za novo zadatu sliku. Problem nije jednostavan pošto slike mogu da sadrže različite varijacije. Tipičan način za rešavanje ovog problema je pristup zasnovan na podacima [5]. Umesto da pokušavamo da opišemo svaku od klasa koju želimo da identifikujemo, koristi se veliki broj slika za svaku od klasa kako bi se izgradio model (klasifikator) koji je u stanju da ih identifikuje. Tradicionalni pristup klasifikaciji je podrazumevao ručno projektovanje različitih ekstraktora vektora obeležja (eng. feature) na osnovu kojih bi se obučavao klasifikator. Međutim, veliki napredak se javio pojavom dubokih KNM i kompletnog obučavanja (eng. end-to-end learning). Duboke KNM imaju sposobnost da izgrađuju hijerarhiju obeležja kroz različite konvolucione slojeve koje poseduju, te kao takve postaju nezamenjive u ulozi ekstraktora obeležja koji uči iz podataka.

Rad je organizovan na sledeći način: poglavlje 2 sadrži opis srodnih radova koji se bave identifikacijom vrsta za potrebe akvatičnog biomonitoringa, kao i javnih skupova podataka koji su korišćeni u ovom radu. U poglavlju 3 dat je opis predloženog metoda uključujući i detalje konkretne implementacije. Ostvareni rezultati i odgovarajuća diskusija dati su u poglavlju 4. Konačno, u zaključku (poglavlje 5) je dat rezime i naznačeni su pravci daljeg istraživanja.

II. Srodni radovi i skupovi podataka

Problem automatizovane taksonomske identifikacije bentonskih makrobeskičmenjaka zasnovane na klasifikaciji slika je obrađen u radovima [6]–[11]. Lytle i dr. [6] su razvili jedan od prvih sistema ove vrste. Njihov sistem BugID koristi Scale Invariant Feature Transform (SIFT) deskriptore [12] u kombinaciji sa Random Forest (RF) klasifikatorom i na STONEFLY9 skupu podataka [6], [13] postiže tačnu klasifikaciju u 95,5% slučajeva.

Larios i dr. [7] su za klasifikaciju koristili tri različita ekstraktora obeležja: Histogram of Oriented Gradients (HOG), Beam Angle Statistics (BAS) i SIFT specijalizovanih za različite delove prostora obeležja. Testiranje predloženog metoda je vršeno na EPT29 skupu podataka koji sadrži 4722

Aleksandar Milosavljević – Elektronski fakultet, Univerzitet u Nišu, Aleksandra Medvedeva 14, 18000 Niš, Srbija (e-mail: aleksandar.milosavljevic@elfak.ni.ac.rs).

Đurađ Milošević – Prirodno-matematički fakultet, Univerzitet u Nišu, Višegradska 33, 18000 Niš, Srbija (e-mail: djuradj@pmf.ni.ac.rs).

Bratislav Predić – Elektronski fakultet, Univerzitet u Nišu, Aleksandra Medvedeva 14, 18000 Niš, Srbija (e-mail: bratislav.predic@elfak.ni.ac.rs).

slike 29 akvatičnih vrsta koje pripadaju redovima *Ephemeroptera*, *Plecoptera* i *Trichoptera* koji se najčešće koriste za procenu stanja akvatičnih ekosistema. Najbolji rezultat od 88,06% je ostvaren korišćenjem Spatial-Pyramid Kernel Support Vector Machines (SVM) klasifikatora u kombinaciji sa stratifikovanom 3-kratnom unakrsnom validacijom.

Kiranyaz i dr. [8] su predložili još jedan klasičan, tj. pre dubokog učenja, pristup morfološkoj identifikaciji makrobeskičmenjaka. Skup podataka korišćen u njihovom istraživanju se sastojao od 1350 slika 8 taksonomskih vrsta. Skup podataka nije javni, te ga nismo koristili u našim eksperimentima. Za ekstrakciju obeležja koristili su ImageJ softver koji generiše 15-to dimenzione vektore obeležja na osnovu kojih su obučavani različiti klasifikatori: SVM, Bayesian Classifiers (BC) i dve neuronske mreže: Multi-Layer Perceptron (MLP) i Radial Basis Function Network (RBFN). Najbolji rezultat i grešku od 3,57% je zabeležio MLP model.

Joutsijoki i dr. [9] su koristili isti skup podataka i metodologiju za ekstrakciju obeležja kako bi ispitali primenjivost veštačkih neuronskih mreža za identifikaciju makrobeskičmenjaka. Eksperimenti su vršeni sa tri arhitekture: MLP, Probabilistic Neural Network (PNN) i RBFN, a MLP se ponovo pokazao najbolje ostvarujući tačnost od 95,3%. Treba napomenuti da je ovde korišćena drugačija metodologija podele skupa, tako da je 80% korišćeno za treniranje, a po 10% za validaciju i testiranje.

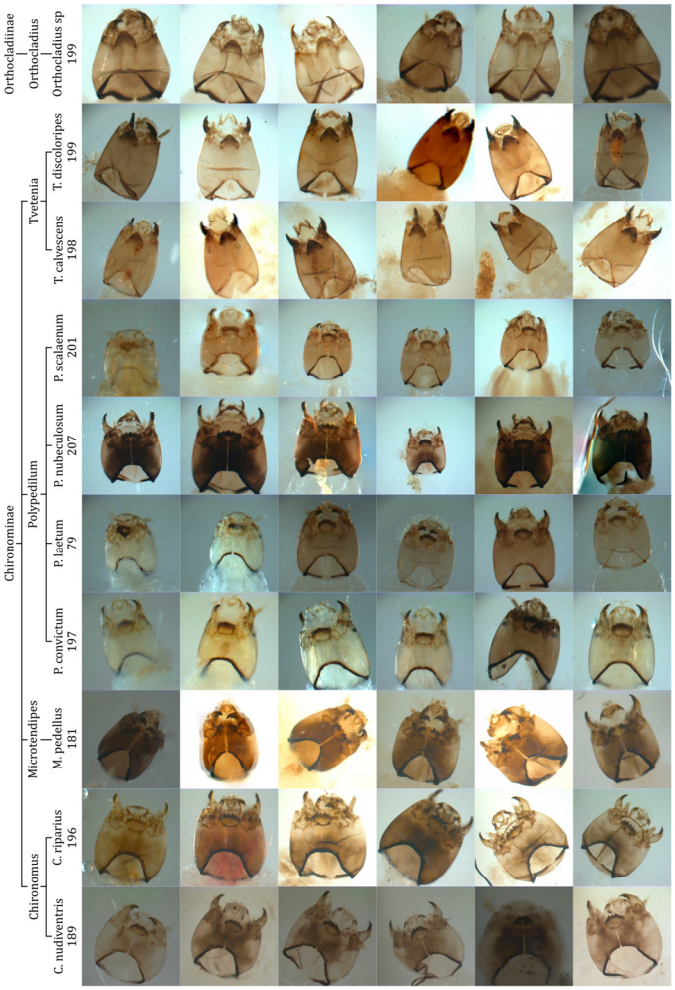
Raitoharju i dr. [10] su kreirali javno dostupan skup FIN-Benthic sa ciljem testiranja različitih metodologija klasifikacije vizuelno sličnih vrsta akvatičnih makrobeskičmenjaka. FIN-Benthic sadrži 3 podskupa sa 64, 29 i 9 vrsta. Broj slika po klasi varira između 7 i 577. U radu je predložena i metodologija za slikanje uzoraka iz različitih uglova korišćenjem dva kamere. Ovo je značajno za proces identifikacije jer omogućava kombinovanje dve nezavisne predikcije pri određivanju vrste uzorka. Još jedna bitna karakteristika FIN-Benthic skupa je da sadrži 10 eksplicitnih podela na trening (50%), validacione (20%) i test (30%) podskupove. Eksperimenti koje su autori sproveli su prvi put iskoristili KNM (AlexNet [14]), kako za ekstrakciju obeležja u kombinaciji sa SVM klasifikatorom, tako i za kompletno obučavanje koje je dalo i nešto bolje rezultate. Najbolje tačnosti po tri definisana podskupa su 75,74% (podskup 1), 81,04% (podskup 2) i 90,14% (podskup 3).

Konačno, rad [11] predstavlja naš doprinos u oblasti automatske identifikacije vrsta larvi hironomida (*Diptera: Chironomidae*). U radu je predstavljen kreirani skup podataka koji sadrži 1846 slika i sastoji se od 10 morfološki vrlo sličnih vrsta iz istog roda ili podfamilije (vidi Sliku 1). U radu je predstavljen metod zasnovan na korišćenju ResNet-50 [15] KNM prethodno obučene na ImageNet [16] skupu koji je dao tačnost od 99,465% na validacionom skupu koji je činio 20% ukupnih podataka.

U Tabeli I prikazani su detalji skupova podataka koji su korišćeni za potrebe istraživanja predstavljenih u ovom radu. Ilustracija CHIRO10 skupa je data na slici 1.

TABELA I
DETALJI KORIŠĆENIH SKUPOVA PODATAKA

Naziv skupa	Pod-skup	Br. klasa	Br. uzoraka	Br. slika	Br. slika po uzorku	Br. slika po klasi
CHIRO10	1	10	1846	1846	1	79–207 (~186)
	2	5				199–684 (~369)
	3	2				199–1647 (~923)
FIN-Benthic	1	64	7705	15074	1–2 (~2)	7–577 (~235)
	2	29	6038	11832	1–2 (~2)	230–577 (~408)
	3	9	1692	3272	1–2 (~2)	322–395 (~363)
STONEFLY9	–	9	774	3845	1–5 (~5)	119–532 (~427)
EPT29	–	28	1608	4794	1–4 (~3)	27–366 (~171)



Sl. 1. Ilustracija CHIRO10 skupa podataka.

III. OPIS METODA

Predloženi metod identifikacije vrsta na osnovu slika se zasniva na dubokom učenju, tj. na obučavanju rezidualne

KNM u ulozi klasifikatora. Da bi se ostvarila robusnost u režimu ograničenog broja trening uzoraka korišćene su sledeće tehnike pri projektovanju i obučavanju klasifikatora:

1. Preneseno učenje (eng. transfer learning)
2. Odbacivanje (eng. dropout)
3. Proširivanje podataka (eng. data augmentation)

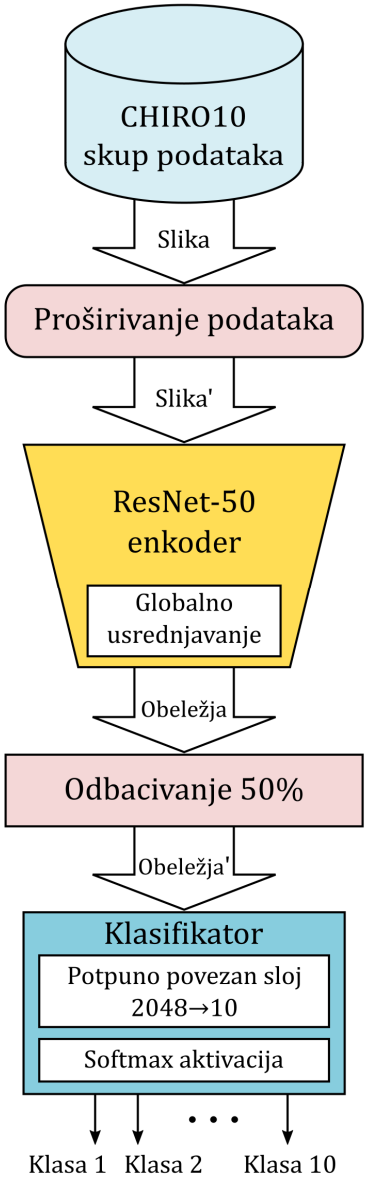
Preneseno učenje predstavlja osnovnu tehniku koja omogućava korišćenje dubokih KNM za rešavanje problema sa relativno malim skupom podataka. Zasniva se na korišćenju prethodno utrenirane mreže na nekom velikom skupu podataka, kao što je npr. ImageNet skup. Treniranje na ImageNet skupu obezbeđuje da mreža izgradi hijerarhiju različitih obeležja koje se mogu naći na fotografijama generalno i koje je pogodno iskoristiti za klasifikaciju novih fotografija. Da bi takav transfer bio moguć, potrebno je zameniti vršni deo mreže zadužen za klasifikaciju i utrenirati ga koristeći niz obeležja koje identifikuje duboka KNM. Uobičajeno je da se deo KNM ispred klasifikatora naziva enkoder. U zavisnosti od prirode novog skupa ovakav pristup može da bude i sasvim dovoljan. Međutim, u našem slučaju ulazni podaci ne predstavljaju nešto što se tipično nalazi na fotografijama, zbog toga je bilo neophodno izvršiti dvofazno obučavanje. Nakon obučavanja klasifikatora u prvoj fazi, u drugoj fazi je vršeno fino podešavanje cele mreže. Fino podešavanje nije ništa drugo do obučavanja celokupne mreže, kako klasifikatora tako i enkodera. Termin fino se koristi da naznači korišćenje vrlo malih koeficijenata učenja kako bi se što manje narušile polazne vrednosti parametara.

Druga tehnika koja je iskorišćena kako bi se povećala robusnost klasifikatora je odbacivanje [17]. Odgovarajući sloj je dodat nakon enkodera, tako da se u svakom koraku obučavanja odbacuje određen procenat (u našem slučaju 50%) obeležja na osnovu kojih se vrši klasifikacija uzoraka. U fazi testiranja i eksploatacije mreže se uzimaju u obzir svi izlazi, ali se vrši skaliranje vrednosti za odgovarajući procenat odbacivanja. Na ovaj način se postiže u proseku isti nivo izlaza koji smo imali kod treniranja. Efekat koji se postiže primenom odbacivanja je da klasifikator mora da se oslanja na više različitih obeležja pri određivanju kategorije. Na ovaj način se izbegava preterano prilagođavanje modela (eng. overfitting), a samim tim i bolji rezultati na validacionom skupu.

Obučavanje neuronske mreže da klasifikuje slike zahteva nalaženje obeležja koje određuju odgovarajuće klase. Taj proces zahteva veliku količinu trening uzoraka kako bi se izolovale ključne karakteristike klasa i dobio klasifikator otporan na različite varijacije koje na slikama mogu da se jave. Kada imamo manju količinu trening podataka, a koristimo model velikog kapaciteta, dešava se model vrlo brzo „zapamti“ sve trening uzorke, ali zato daje loše rezultate na validacionom skupu. Tipično korišćena tehnika koja služi da se ovo izbegne je proširivanje podataka. Proširivanje podataka podrazumeva primenu nasumičnih transformacija nad ulaznim slikama tako da se u svakom trening ciklusu mreži predoči nešto što ranije nije „videla“. U zavisnosti od prirode skupa podataka, tipične transformacije uključuju obrtanje (eng. flip), rotaciju, translaciju, skaliranje, zakošavanje, promenu osvetljaja i kontrasta, itd.

A. Arhitektura mreže

Vodeći se prethodno obrazloženim principima, za klasifikaciju je usvojena arhitektura zasnovana na ResNet-50 [15] enkoderu prikazana na slici 2.



Sl. 2. Šematski prikaz korišćene arhitekture zasnovane na ResNet-50 mreži.

Na izbor ResNet-50 mreže utrenirane na ImageNet skupu kao enkodera je uticalo nekoliko faktora: dobri rezultati na ImageNet skupu, veličina mreže u pogledu broja parametara, memorijsko zauzeće u toku treniranja, brzina obučavanja, kao i dostupnost modela u korišćenom programskom okruženju. ResNet arhitektura generalno vrlo često predstavlja dobar inicijalni izbor zbog dobrog odnosa preciznosti i brzine obučavanja.

ResNet-50 enkoder poseduje 23.587.712 parametara, dok na izlazu daje 2048 obeležja. Izlazi se dobijaju korišćenjem

globalnog usrednjavanja (eng. *global average pooling*) po izlaznoj mapi obeležja koja za korišćeni ulaz dimenzija 512x512 piksela iznosi 16x16x2048.

Na dobijenih 2048 izlaza iz enkodera se primenjuje odbacivanje sa faktorom 50%. Ovo u praksi znači da se u fazi obučavanja polovina, tj. 1024 nasumično izabranih izlaza postavi na nulu. Tako modifikovani izlazi enkodera se dovode na potpuno povezani sloj sa 3 neurona gde svaki izlaz odgovara jednoj klasi. Sloj poseduje 6.147 parametara koji se obučavaju u prvoj fazi. Na izlaze se primenjuje *softmax* aktivaciona funkcija (1) čime ovaj sloj dobija ulogu klasifikatora obeležja koje daje ResNet-50 enkoder. Izlazi softmax funkcije predstavljaju verovatnoće da je tekući uzorak pripadnik neke od klasa. Ovo praktično znači da pojedinačni izlazi imaju vrednosti iz intervala [0, 1] i da je zbir svih izlaza jednak 1.

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}. \qquad (1)$$

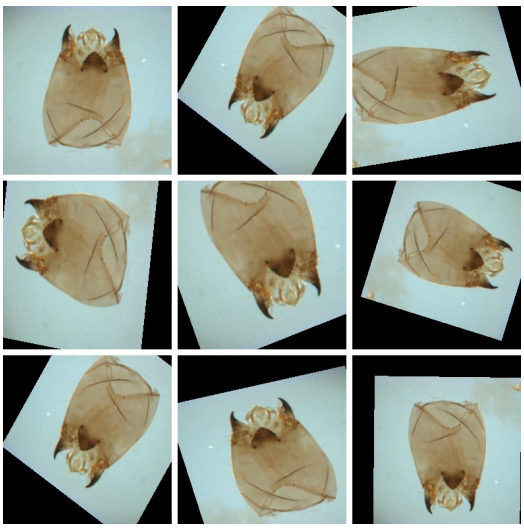
B. Implementacija i obučavanje mreže

Za implementaciju predložene arhitekture iskorišćen je programski jezik Python i biblioteka Keras [18]. Keras je biblioteka visokog nivoa koja definiše pojednostavljeni interfejs za implementaciju dubokih neuronskih mreža i u našem slučaju se oslanja na TensorFlow [19] biblioteku za realizaciju svih funkcionalnosti.

U okviru *applications* modula Keras poseduje nekoliko dubokih KNM arhitektura istreniranih na ImageNet skupu. Instanciranjem klase *ResNet50*, uz odgovarajuće parametre, dobijamo enkoder za naš model. Na izlaz enkodera se nadovezuje *Dropout* i *Dense* sloj sa *softmax* aktivacijom čime se dobija kompletan model. Kako se u prvoj fazi obučavanja težine ResNet-50 enkodera ne menjaju, potrebno je za sve konvolucione slojeve u enkoderu postaviti atribut *trainable* na *False*.

Kreirani model je kompajliran tako da koristi *Adam* [20] algoritam za optimizaciju (*optimizers* modul), *sparse_categorical_crossentropy* tip greške (*losses* modul), dok se kao metrika tačnosti koristi *sparse_categorical_accuracy* (*metrics* modul). Algoritam optimizacije je izabran zbog brze konvergencije, dok su greška za obučavanje i odgovarajuća metrika tačnosti standardni izbor za problem klasifikacije. Varijante ovih funkcija sa prefiksom *sparse_* se koriste kada se klase koje predstavljaju očekivani izlaz zadaju kao celi broj (0, 1 ili 2 u našem slučaju).

Za potrebe proširivanja podataka iskorišćena je Keras ugrađena klasa *ImageDataGenerator* koja se nalazi u *preprocessing* modulu, podmodul *image*. Pri konstruiranju odgovarajućeg generatora slika definišu se opsezi za različite transformacije koje će nasumično primenjivati. U našem slučaju korišćeno je horizontalno i vertikalno obrtanje slike, rotacija do ±90°, translacija do ±15% po oba pravca, promena osvetljaja do ±20%, zakošavanje i skaliranje do ±10%. Ilustracija je data na slici 3. Proširivanje podataka se vrši samo za trening skup, do za potrebe validacije koriste neizmenjene slike.



Sl. 3. Ilustracija proširivanja podataka. Prva slika (gore-levo) predstavlja ulaz, dok su ostale slike nastale primenom nasumičnih transformacija.

S obzirom da je korišćen generator slika, za obučavanje mreže koristi se metod *fit_generator*. Dodatna kontrola procesa obučavanja je u Kerasu moguća prosleđivanjem liste *callback* objekata. Odgovarajuće klase se nalaze u modulu *callbacks* i u našem slučaju iskorišćene su: *LearningRateScheduler*, *EarlyStopping*, *ModelCheckpoint* i *CSVLogger*.

LearningRateScheduler obezbeđuje definisanje proizvoljne funkcije za izmenu koeficijenta brzine obučavanja u zavisnosti od tekuće epohe obučavanja. U našem slučaju, ovaj *callback* je iskorišćen za implementaciju tzv. kosinusnog kaljenja (eng. *cosine annealing*) [21]. Kod kosinusnog kaljenja koeficijent obučavanja se smanjuje po kosinusnoj funkciji od neke inicijalne do neke minimalne vrednosti u toku određenog broja epoha koje su definisane periodom ponavljanja. U prvoj fazi obučavanja je korišćena perioda ponavljanja od 10 epoha, sa inicijalnim koeficijentom obučavanja u rasponu od 10^{-3} do 10^{-5}, uz smanjivanje 0,7 puta pri svakoj novoj periodi.

Kako ime sugeriše, *EarlyStopping callback* se koristi za ranije zaustavljanje procesa obučavanja ukoliko u određenom broju epoha nema napretka po određenom parametru. U našem slučaju je korišćeno 30 epoha i praćena je tačnost na validacionom skupu.

CSVLogger callback se koristi za snimanje greški i tačnosti nad trening i validacionim skupom u toku procesa obučavanja, a u cilju kasnije vizuelizacija ovog procesa.

Konačno, *ModelCheckpoint* je iskorišćen za snimanje najboljeg rezultata u pogledu tačnosti postignute nad validacionim skupom. Ovako zabeležen model je iskorišćen za narednu fazu obučavanja, tj. fazu finog podešavanja.

Fino podešavanje je vršeno na gotovo identičan način, uz par sitnijih izmena. Nakon učitavanja modela dobijenog iz

prve faze obučavanja, izvršeno je aktiviranje obučavanja za sve slojeve iz ResNet-50 enkodera (*trainable* atribut postavljen na *True*). Da bi se izbegla drastična izmena težina u enkoderu, koeficijent obučavanja se kretao u rasponu od 10^{-5} do 10^{-7}. Konačno, da bi se dobio model koji daje najbolje rezultate na celokupnom skupu podataka, umesto standardnog validacionog, iskorišćen je kompletan neizmenjen skup podataka. Obučavanje je i dalje rađeno sa proširivanjem podataka trening skupa. Odgovarajući rezultati su prezentovani u narednom poglavlju.

IV. Rezultati i diskusija

U cilju evaluacije predloženog metoda, za svaki skup podataka, obučili smo 10 modela, po jedan za svaku podelu. Rezultati su dobijeni evaluacijom obučenih modela na odgovarajućem test skupu koji sadrži 30% svih uzoraka iz skupa podataka. Treba naglasiti da slike iz trening skupova nisu ni na koji način korišćene u procesu obučavanja modela, te na taj način predstavljaju realne očekivane performanse modela na nepoznatim podacima.

S obzirom da svi skupovi podataka, sem našeg CHIRO10 skupa, poseduju više slika po uzorku, kao metrika tačnosti je izabrana tačnost na nivou uzorka, a ne slike. Da bi uporedili kako različiti metodi akvizicije više slika po uzorku utiču na rezultat, računali smo i tačnost na nivou slike. Tačnost na nivou uzorka je računata usrednjavanjem predikcija, tj. raspodela verovatnoća, dobijenih za svaku od slika uzorka, te dodelom najverovatnije klase. Na primer, ukoliko imamo 3 klase i 2 slike za uzorak, ukoliko model prediktuje (60%, 10% i 30%) za prvu sliku i (10%, 40% i 50%) za drugu sliku, tada se dobija srednja predikcija od (35%, 25% i 40%), te će uzorak biti pridružen trećoj klasi.

TABELA II
SREDNJA TAČNOST I STANDARDNA DEVIJACIJA KLASIFIKACIJE UZORAKA

Naziv skupa	Podskup	Srednja tačnost [%]	Standardna devijacija [%]
CHIRO10	1	96,79	0,70
	2	99,17	0,36
	3	99,33	0,37
FIN-Benthic	1	81,00	0,85
	2	85,64	0,70
	3	96,58	0,70
STONEFLY9	–	99,01	0,61
EPT29	–	97,43	0,49

TABELA III
POREĐENJE SA ORIGINALNIM REZULTATIMA I REZULTATIMA NA NIVOU SLIKE

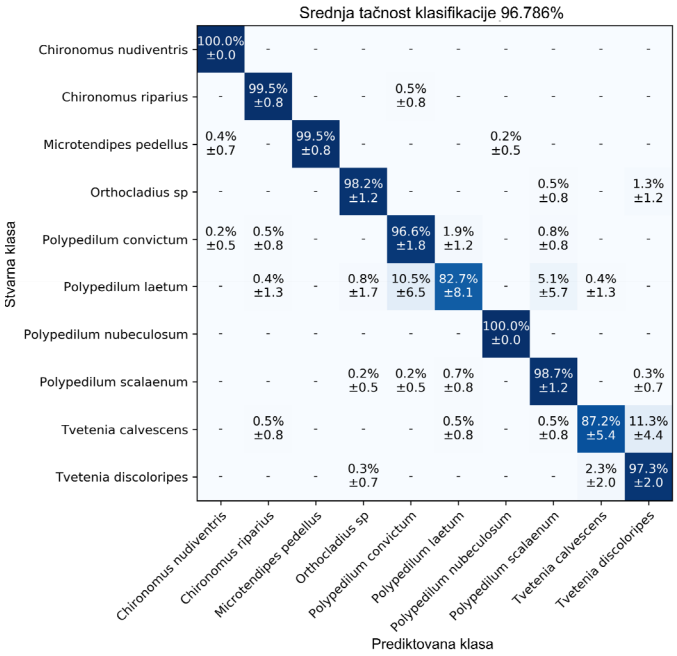
Naziv skupa	Pod-skup	Origin. rezultati [%]	Naši rezultati [%]	Pobo-ljšanje [%]	Naši rezultati po slici [%]	Pobo-ljšanje po uzorku [%]
FIN-Benthic	1	75,74	81,00	+5,29	76,59	+4,41
	2	81.04	85,64	+4,60	81,19	+4,45
	3	90,14	96,58	+6,44	93,63	+2,95
STONE-FLY9	–	94,50	99,01	+4,55	97,69	+1,32
EPT29	–	88,06	97,43	+9,56	95,37	+2,06

U Tabeli II prikazani su usrednjeni rezultati koji prikazuju tačnost na nivou uzorka kao i odgovarajuća standardna devijacija. U Tabeli III prikazano je poređenje dobijenih rezultata korišćenjem predloženog metoda i originalnih rezultata (FIN-Benthic [10], STONEFLY9 [6] i EPT29 [7]).

Predloženi metod je pokazao značajno poboljšanje u identifikaciji vrsta na svim javnim skupovima podataka koje se kreće u opsegu od 4,55 do 9,56%. Jedini lošiji rezultat je zabeležen na našem CHIRO10 skupu, ali razlog za to je drugačija strategija evaluacije rezultata koja koristi dvostruku unakrsnu evaluaciju, tj. odvojeni validacioni i test skup. Takođe razlog za lošiji rezultat leži u činjenici da su postojeći eksperimenti vršeni sa slikama 256x256 piksela, dok je originalni rezultat postignut sa slikama 512x512 piksela.

Ukoliko pogledamo matricu konfuzije prikazanu na slici 4, uočava se da je najlošiji rezultat od 82,7±8,1% ostvaren kod klasifikacije *Polypedilum laetum* jedinki. Razlog za to treba tražiti u činjenici da sve ostale klase poseduje ~200 slika po klasi, dok za ovu klasu imamo svega 79 slika. Klasa *Tvetenia calvescens* se nalazi na drugom mestu sa 87,2±5,4%, međutim ovaj put je većina pogrešno klasifikovanih uzoraka (11,3±4,4%) otišla na klasu *Tvetenia discoloripes* koja je vrlo slična (pripada istom rodu).

Uticaj više slika po uzorku na tačnost klasifikacije uzoraka je pokazalo poboljšanje od 1,32 do 4,45%. Pokazuje se da metod akvizicije slika korišćen kod FIN-Benthic skupa, gde dve kamere slikaju uzorak iz dva ugla, daje bolje rezultate od odgovarajućih metoda korišćenih kod STONEFLY9 i EPT29 skupova podataka gde je broj slika po uzorku 5 i 3, ali se koristi jedna kamera.



Sl. 4. Matrica konfuzije za CHIRO10 skupa podataka (podskup 1).

Konačno, kvalitet slika takođe igra vrlo bitnu ulogu. Ukoliko uporedimo FIN-Benthic podskup 2 (29 klasa, 6038 uzoraka i 11832 slika) sa EPT29 skupom (28 klasa, 1608

uzoraka i 4794 slika) svi parametri govore u korist FIN-Benthic skupa. Međutim, tačnost FIN-Benthic podskup 2 skupa iznosi 85,64%, dok je tačnost kod EPT29 skupa 97,43%. Razlog za ovo je, po našem mišljenju, kvalitet slika koji se ogleda u nivou detalja. Kod FIN-Benthic skupa je taj kvalitet mnogo lošiji nego kod ostalih skupova što se verovatno ogleda i na tačnost klasifikacije.

V. Zaključak

U radu je predložen i evaluiran pristup za identifikaciju vrsta za potrebe akvatičnog biomonitoringa. Predloženi metod se oslanja na tri tehnike dubokog učenja koje imaju za cilj poboljšanje robusnosti kada se obučavanje vrši na relativnom malim skupovima podataka: preneseno učenje, proširivanje podataka i odbacivanje. Preneseno učenje je primenjeno korišćenjem ResNet-50 KNM prethodno obučene na ImageNet skupu podataka. Za evaluaciju je iskorišćen naš CHIRO10 skup podataka, kao i nekoliko javnih skupova (FIN-Benthic, STONEFLY9 i EPT29).

Da bi mogli da poredimo rezultate na različitim skupovima podataka, izvršili smo unifikaciju trening procesa korišćenjem slika veličine 256x256 piksela i 10 podela podataka kako bi izmerili srednju tačnost i standardnu devijaciju. Rezultati su pokazali značajna poboljšanja u odnosu na originalne radove, potvrdili značajan uticaj korišćenja više slika po uzorku, ali i pokazali da se broj uzoraka po klasi mora dobro izvagati.

Zahvalnica

Prikazani rezultati dobijeni su u okviru istraživanja na projektima III-43007 i III-47003 koje finansira Ministarstvo prosvete, nauke i tehnološkog razvoja Republike Srbije.

Literatura

[1] IPBES, "Summary for policymakers of the global assessment report on biodiversity and ecosystem services of the Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services. S. Díaz, J. Settele, E. S. Brondizio E.S., H. T. Ngo, M. Guèze, J. Aga," Bonn, 2019.

[2] P. F. M. Verdonschot, "Evaluation of the use of Water Framework Directive typology descriptors, reference sites and spatial scale in macroinvertebrate stream typology," *Hydrobiologia*, vol. 566, no. 1, pp. 39–58, Aug. 2006.

[3] G. W. Hopkins and R. P. Freckleton, "Declines in the numbers of amateur and professional taxonomists: Implications for conservation," *Anim. Conserv.*, vol. 5, no. 3, pp. 245–249, Aug. 2002.

[4] F. C. Jones, "Taxonomic sufficiency: The influence of taxonomic resolution on freshwater bioassessments using benthic macroinvertebrates," *Environ. Rev.*, vol. 16, no. NA, pp. 45–69, Dec. 2008.

[5] "CS231n Convolutional Neural Networks for Visual Recognition: Image Classification." [Online]. Available: http://cs231n.github.io/classification/. [Accessed: 23-Dec-2019].

[6] D. A. Lytle *et al.*, "Automated processing and identification of benthic invertebrate samples," *J. North Am. Benthol. Soc.*, vol. 29, no. 3, pp. 867–874, 2010.

[7] N. Larios *et al.*, "Stacked spatial-pyramid kernel: An object-class recognition method to combine scores from random trees," in *2011 IEEE Workshop on Applications of Computer Vision, WACV 2011*, 2011, pp. 329–335.

[8] S. Kiranyaz *et al.*, "Classification and retrieval on macroinvertebrate image databases," *Comput. Biol. Med.*, vol. 41, no. 7, pp. 463–472, 2011.

[9] H. Joutsijoki *et al.*, "Evaluating the performance of artificial neural networks for the classification of freshwater benthic macroinvertebrates," *Ecol. Inform.*, vol. 20, pp. 1–12, Mar. 2014.

[10] J. Raitoharju *et al.*, "Benchmark database for fine-grained image classification of benthic macroinvertebrates," *Image Vis. Comput.*, vol. 78, pp. 73–83, Oct. 2018.

[11] D. Milošević *et al.*, "Application of deep learning in aquatic bioassessment: Towards automated identification of non-biting midges," *Sci. Total Environ.*, vol. 711, p. 135160, Apr. 2020.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[13] G. Martinez-Munoz *et al.*, "Dictionary-free categorization of very similar objects via stacked evidence trees," 2010, pp. 549–556.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, vol. 2, pp. 1097–1105.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 770–778.

[16] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[17] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arxiv.org*, 2012.

[18] "Keras: The Python Deep Learning library." [Online]. Available: https://keras.io. [Accessed: 24-Jan-2020].

[19] "TensorFlow: An end-to-end open source machine learning platform." [Online]. Available: https://www.tensorflow.org/. [Accessed: 24-Jan-2020].

[20] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference for Learning Representations*, 2015.

[21] J. Jordan, "Setting the learning rate of your neural network," 2018. [Online]. Available: https://www.jeremyjordan.me/nn-learning-rate/. [Accessed: 24-Jan-2020].

Abstract

Aquatic insects and other benthic macroinvertebrates are mostly used as bioindicators of the ecological status of freshwaters. However, an expensive and time-consuming process of species identification represents one of the key obstacles for reliable biomonitoring of aquatic ecosystems. In this paper, we proposed a deep learning-based method for species identification that we evaluated on several available public datasets (FIN-Benthic, STONEFLY9, and EPT29) along with our CHIRO10 dataset. The proposed method relies on three deep learning techniques used to improve robustness when training is done on a relatively small dataset: transfer learning, data augmentation, and feature dropout. The results for all datasets were obtained using 256×256 images and averaging on 10 data splits in training (50%), validation (20%), and test (30%) sets. The results show significant improvement compared to original contributions and confirms that there is a considerable gain when there are multiple images per specimen.

Species identification for aquatic biomonitoring using convolutional neural networks and deep learning

Aleksandar Milosavljević, Đurađ Milošević and Bratislav Predić