IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2017.DOI*

# Data-driven Intelligence System for General Recommendations of Deep Learning Architectures

**GJORGJI NOVESKI[1], TOME EFTIMOV [2], KOSTADIN MISHEV [1], and MONIKA SIMJANOSKA [1]**

[1]Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, Rugjer Boshkovikj, 16, 1000 Skopje, N. Macedonia (e-mail: gjorgji.noveski@students.finki.ukim.mk, {kostadin.mishev,monika.simjanoska}@finki.ukim.mk)

[2]Computer Systems Department, Jozef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia (e-mail: tome.eftimov@ijs.si)

Corresponding author: Gjorgji Noveski (e-mail: gjorgji.noveski@students.finki.ukim.mk).

This work was supported by the Slovenian Research Agency (research core funding programmes P2-0098 and project Z2-1867).

**ABSTRACT** Choosing optimal Deep Learning (DL) architecture and hyperparameters for a particular problem is still not a trivial task among researchers. The most common approach relies on popular architectures proven to work on specific problem domains led on the same experiment environment and setup. However, this limits the opportunity to choose or invent novel DL networks that could lead to better results. This paper proposes a novel approach for providing general recommendations of an appropriate DL architecture and its hyperparameters based on different configurations presented in thousands of published research papers that examine various problem domains. This architecture can further serve as a starting point of investigating DL architecture for a concrete data set. Natural language processing (NLP) methods are used to create structured data from unstructured scientific papers upon which intelligent models are learned to propose optimal DL architecture, layer type, and activation functions. The advantage of the proposed methodology is multifold. The first is the ability to eventually use the knowledge and experience from thousands of DL papers published through the years. The second is the contribution to the forthcoming novel researches by aiding the process of choosing optimal DL setup based on the particular problem to be analyzed. The third advantage is the scalability and flexibility of the model, meaning that it can be easily retrained as new papers are published in the future, and therefore to be constantly improved.

**INDEX TERMS** Deep Learning, Intelligent system, Hyperparameters selection, DL architecture selection, Multi-label classification.

## I. INTRODUCTION

Building a DL intelligent model is still a non-deterministic and challenging process. Most of the difficulties arise from the need of sufficient empirical information to aid the decision of the architecture type and hyperparameters to be used in the DL network. This information is usually obtained experimentally, meaning days, weeks and even months are spent on training models that might fail to produce the desired results. Therefore, the researchers often choose DL solutions that are similar to the problem at hand. Even though it does not consistently achieve the best results, it is still considered a good starting point.

Recognizing the starting point becomes more complicated as many novel research papers are published each day. Even more, those research papers might contain some "hidden" information that is crucial for adapting a particular DL to the problem analyzed. Reading and capturing those minor differences in the papers is nearly impossible. The modern NLP methods can be applied to extract structured knowledge from unstructured scientific data and then use the knowledge to train intelligent models that are able to provide directions on the optimal starting point for choosing DL architecture and parameters for the particular problem.

The representation of the problem domain is also a challenging task. The nature and the size of the data are one of the most important aspects to consider when providing an appropriate experimental setup. Tackling the problem of choosing a good DL architecture comes in many forms, from implementing new forward propagation techniques [8] that provide more stable architectures to building "umbrella" models which work on a broad range of domains [9], [10]. Another way to build effective neural network models is to

| Paper | Description | Results |
|---|---|---|
| [1] | Design CNN architectures by using hill climbing strategy coupled with network morphism and cosine annealing | Error rate on CIFAR10:4.4%, CIFAR100:19.6% |
| [2] | Propose Adversarial NAS, method for designing both generator and discriminator architecture in generative adversarial networks (GANs) | Inception Score on CIFAR10:8.74 |
| [3] | Morph existing trained networks to new ones while preserving network function by proposing parametric-activation functions. (handles changes in depth,width, kernel size) | Accuracy on CIFAR10: 84% |
| [4] | Utilizing regression algorithms and sparse optimization methods to create a child network with an extra layer that is similar to its parent network | Accuracy on VGG16:84%, VGG17:85%, VGG19:89% |
| [5] | Exploring architecture space using a reinforcement learning agent which grows a network. Network architecture is encoded and certain network transformations are taken | Error rate on CIFAR10+:4.23% |
| [6] | Adaptively choosing a hyper-parameter configuration while allocating iterations, samples and features | Error rate CIFAR10: 17% |
| [7] | Using sparse recovery and comprehensive sensing to solve HPO and NAS problems | Error rate CIFAR10:2.74% |

TABLE 1: Overview of related work.

use Neural architecture search (NAS) [11], [12] alongside with hyper-parameter optimization (HPO) [13], [14]. These techniques automate the process of designing a neural network which in turn help researchers make use of models with comparable or better results than their state-of-the-art counterparts [15]. An issue that might not be so evident while deploying DL architectures is dataset shift. Differences between the distributions of the training and the test set is a variable that should be taken into account when building DL models. From [16] we can see that out-of-distribution (OOD) inputs have a significant impact on a model's performance and uncertainty. Being able to quantify this uncertainty using specific methods [17], [18] helps practitioners have some insight into the models' performances during its lifetime. Under high uncertainty, a model can refrain from making a decision in order to mitigate potential problems it may cause in its environment. Several threats can arise when a system makes wrongful decisions [19], nevertheless, promising advances are made that warrant early detection of poor predictions [20].

This paper describes a comprehensive methodology that surpasses the preprocessing difficulties regarding the format of the scientific papers, the unstructured data, and the missing information to create a new database. This new database built from structured data can be constantly updated with novel research papers and, therefore, can be used to develop intelligent models that are fed with keywords describing the problem domain. As the researcher describes the problem domain by choosing appropriate keywords, the intelligent models predict the most suitable DL architecture, layer type, and activation functions as starting experimental setup that is expected to produce satisfying results. Since there is no dataset that can be utilized to provide general recommendations about which architecture and hyperparameters should be used based on the keywords that describe the paper, first we annotate textual data that is used for training a NER model. To create the annotated corpus, a rule-based method was used. The first NER model provides us with some silver standard that is further used to fine tune predefined corpus-based NER models (e.g., specialized models). Using the NER models, we are able to create a dataset using information about architecture type, activation function, and building blocks. Such a dataset can be further used to train ML models that provide recommendations.

These predictions are in the form of general recommendations (e.g., Given an input vector of keywords: signal processing, threshold prediction, real time, the trained model provides the following recommendation: Use CNN architecture type with ReLU activation function and a convolutional layer alongside pooling layers as building blocks. The outcome of the intelligent system can also be further tuned based on the characteristics of the data set that is being solved, since it has already provided architecture that is utilized for the same problem. Those models adapt the concept of online training by continuously updating their knowledge with the latest advances in published research papers that present updates in model configuration, architecture, and hyperparameters to achieve better results in the same tasks.

The rest of the paper is organized as follows. Section II presents similar efforts from the competing papers as well as an appropriate comparison with the benefits from the proposed approach in this paper. The detailed methodology that includes the database creation and the intelligent models training is provided in Section III. The results are presented in Section IV. A comprehensive discussion is provided in Section V and the final Section VI presents a conclusion on the performance of the intelligent models as well as directions for future improvements.

## II. RELATED WORK

Looking at network architecture search (NAS) solutions, [21] manage to overcome the problem of designing a neural net by using reinforcement learning (RL). They use an RNN to generate descriptors of candidate networks and train them by RL, achieving better results than previous state-of-the-art models. On image classification, evolutionary algorithms might be used to find an architecture that performs well. Other solutions such as the one proposed in [22] utilize many workers that compare trained architectures, picking ones with

higher validation scores and mutating them by adding or removing layers. They report only a few restrictions on the mutation operations allowing for a large search space. Other NAS techniques are able to achieve fast convergence on their models with fewer parameters resulting in a smaller memory footprint and fewer GPU train times [23], [24]. The rest of the notable related work together with accompanying results is provided in Table 1. Some of the works are designed to output a trained model for only a specific type of architecture [1], [2], [25], while others are general systems that seek to improve existing Machine learning models [3], [26]. None of the efforts gave a primary focus on the data they process and instead try to transform architectures, create specific objective functions, minimize known ones, or other approaches.

We address this problem by proposing a data-driven system, extracting information from thousands of scientific papers, representing the effort the scientific community is taking into building intelligent systems. The system is a novel method for learning deeper meaning, taking into account scientific text. Instead of being restricted to a particular field, our method can be used to learn different representations in the text depending on the need. Capturing the essence of a scientific paper are its keywords, so only by using the "Keywords" section of a paper, the model can predict what kind of architecture and hyperparameters should be used to get good starting results. With the further development in the research field of DL, systems designed for specific tasks are constantly improving, e.g., CNNs are used for object detection and computer vision [27], [28], RNNs for NLP and speech recognition [29], etc. We aim to unify hidden knowledge in unrelated scientific papers and make building DL intelligent systems easier, thus reducing the number of models that need to be tested. Our effort will help researchers pick a starting approach in building their DL intelligent system.

## III. THE METHODOLOGY

The proposed methodology is comprised of two parts. The first is the methodology developed to create the database, and the second is the methodology for creating the DL intelligent recommendation models.

### A. CREATING THE DATABASE

A large corpus of scientific papers is required to achieve a good intelligent system for recommending a DL architecture. Since such corpus has not been previously created, the database was built from scratch containing a total of 19,868 scientific papers. All of the collected papers were related to Machine Learning and Deep Learning. A portion of them are studies done in Computer Science, while the remaining portion are papers where their methodologies are applied to solve some specific applications (e.g., social humanities, chemistry, biology, etc.).

Extracting relevant data from a text can be considered a separate research problem that requires cautiously choosing appropriate preprocessing methods. From the currently known solutions for information extraction, there are various methods for accomplishing that task because each method depends on the context of the text that is being processed (encyclopedic text, scientific papers, general web text, etc.). The methods can be categorized in two categories: metadata extraction [30], [31] and key-insights extraction [32]. Metadata focuses on general information around a scientific paper, like title, authors, and publication date, while key insights is information that is obtained from the body of the paper.

The methodology established in this research to extract information from the scientific papers is based on a key-insights approach and is described in Figure 1. The steps until "Training specialized NER models" are consider part of the NER pipeline, subsequently the following steps are responsible for creating the dataset. A NER approach is chosen because of it's applicability to our problem and since using a rule-based approach to extracting entities is not as robust. We later see how the NER models do in fact give good results.

Unfortunately, the extraction of textual data from PDFs by using some common libraries led to many tables, images, headers, footers, and other elements in the PDF document being undesirably extracted as textual data. Thus, the result is corrupted sentences and added noise. To avoid this problem, some preprocessing is performed and the database was created in eight consequential steps described as follows.

- **Choosing optimal DL sentences**: Manual annotations have been done in order to create an annotated corpora that can be used for training the "specialized" NER models. For reproducible results all annotated corpora are publicly available on our GitHub repository. The selection of the sentences has been done in order to have as much of a uniform distribution as possible of the concepts and to manage imbalanced data during training of the models. Diverse DL terms were included, such as: hidden layers, transfer learning, model, hyperparameters, learning rate, CNN, weights, parameters, etc. The annotated dataset is available at: https://github.com/Gjorgji-Noveski/Intelligent-System-for-general-recommendations. Here are two examples with annotated named entities in bold:
  - "We apply **deep learning methods** in this paper, namely we use **convolutional neural networks** (**CNNs**) for description and prediction of the red blood cells' trajectory, which is crucial in modeling of a blood flow."
  - "The **Dropout** operation with probability p = 0.5 is used to prevent **overfitting** during the **training phase**."
- **Training a "general" NER model**: We refer to this model as a "general" NER model because it is trained to detect sentences containing various named entities from the field of Deep Learning. The training data is carefully selected because we believe having hand-picked sentences with entity annotations will provide a robust

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3124633, IEEE Access



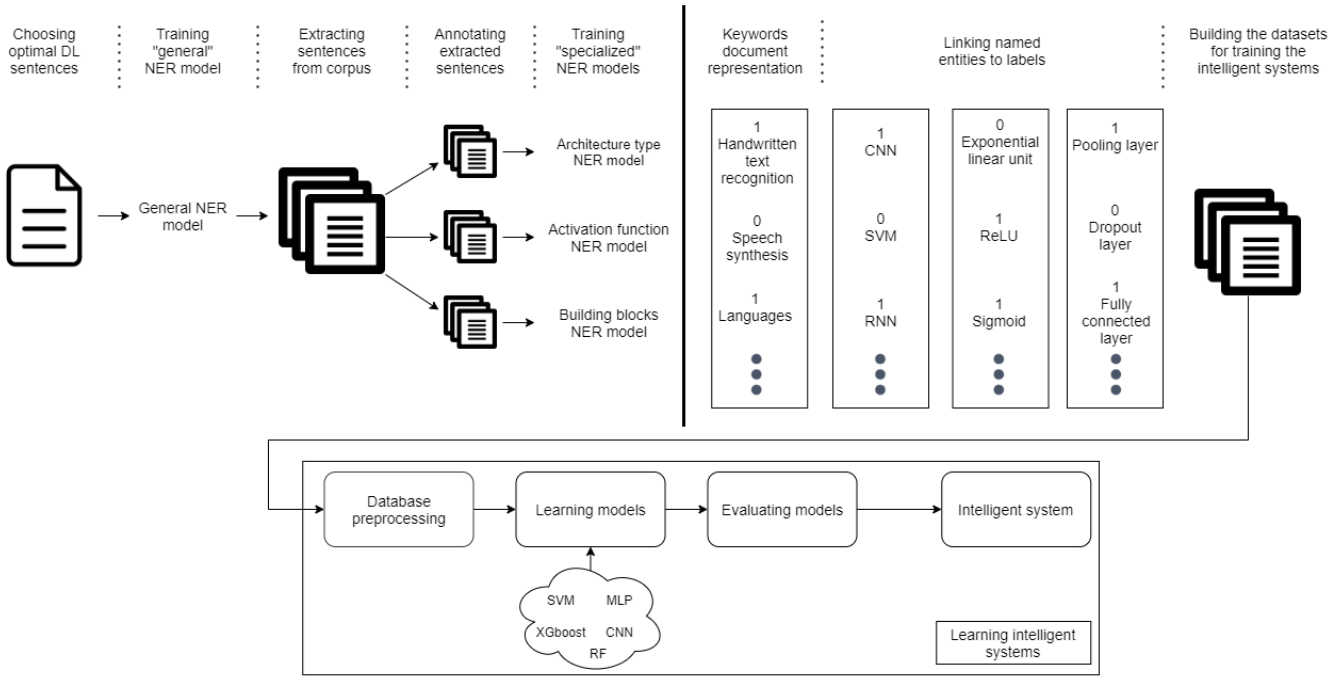



FIGURE 1: Database creation and model learning process

| Architecture type | | Activation functions | Building blocks |
|---|---|---|---|
| Perceptron | Hopfield network | Linear activation function | Fully connected layer |
| Feed forward | Boltzmann machine | Rectified linear unit | Recurrent layer |
| Radial basis network | Deep belief network | Sigmoid | Pooling layer |
| Recurrent neural network | Convolutional neural network | Gaussian error linear unit | Convolutional layer |
| Long short-term memory | Generative adversarial network | Softmax | Deconvolutional layer |
| Gated recurrent unit | Liquid state machine | Hyperbolic tangent function | Dropout layer |
| Autoencoder | Extreme learning machine | Softsign function | Softmax layer |
| Neural turing machine | Echo state network | Exponential linear unit | Subsampling layer |
| Deconvolutional neural network | Deep residual network | Leaky rectified linear unit | GRU layer |
| Convolutional inverse graphics network | Kohonen network | | |
| Probabilistic neural network | Support vector machine | | |
| | Markov chain | | |

TABLE 2: Labels per dataset

| Architecture type | Activation functions | Building blocks |
|---|---|---|
| Long short-term memory Support vector machine | Rectified linear unit | Fully connected layer |
| Perceptron + Feed forward | Sigmoid | Pooling layer |
| Recurrent neural network + Gated recurrent unit | Softmax | Convolutional layer |
| Autoencoder + Generative adversarial network | Hyperbolic tangent function | Softmax layer |
| Markov chain + Boltzmann machine | | |

TABLE 3: Labels per dataset after filtering labels that occur less than 300 times

NER model that will be able to find the required named entities. The model that we used for training was a pre-trained one, namely using "en_core_web_sm" from the "SpaCy" library [33] which offered an already trained NER model on English written text. In the core a deep convolutional neural network with residual connections is used. After training it is evaluated on a test set and achieved f1-score of 0.77.

- **Extracting sentences from corpus**: The "general" NER model is run on the whole corpus of scientific papers, which returns sentences that contain named entities.
- **Annotating extracted sentences**: From the newly acquired sentences, words are annotated in a way to be used to train three new NER "specialized" models covering architecture type, activation functions and building blocks (DL layers). We have decided to train separate NER models for different types of entities, since in this paper, we are not looking into the relations that exist between the entities. Having separate NER models for them can also provide more robust results since they are trained only on a specific problem landscape. These three categories were chosen because of their prevalence in numerous scientific papers in which DL solutions are deeply explained. There was no part-of-speech tag-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3124633, IEEE Access



ging involved since the morphological information is irrelevant to us, simply a rule-based method was used to annotate the sentences. Having an annotated domain specific dataset provides better results as was shown in [34]. Table 2 presents the content of the Architecture type, Activation functions and Building blocks categories.

- **Training "specialized" NER models**: Even though spaCy offers a pretrained model, that model recognizes only general entities (such as person, organization and location) and needs to be fine tuned on a specific domain. Our spaCy NER (i.e., general model) first allows us to create a silver annotated corpus, on which the pretrained corpus-based spaCy NER models are further fine tuned (i.e., specialized models). Same as the "general" NER model, the "en_core_web_sm" model was used for fine tuning. The evaluation of these models on a separate test set yielded f1-score of **0.99** for the architecture type NER model, **0.99** for the activation function NER model and **0.97** for the building blocks NER model.
- **Keywords document representation**: The main goal in the research is to achieve an intelligent recommendation system able to propose optimal configurations from the three categories, based on a user's keywords input. Thus, each scientific paper is preprocessed to extract its keywords, representing the input vector features. The binary values of the features represent whether the keyword is present in the relevant scientific paper. A total of 16,123 unique keywords were identified.
- **Linking named entities to labels**: The "specialized" models are run on the whole corpus to check for named entities. When a match is found, the vector containing the keywords (the input) along with a vector of the corresponding named entities (the targets to be predicted) is extracted.
- **Building the datasets for training the intelligent systems**: At the final step, a dataset of three distinct subsets has been created. The first is comprised of *Keywords vectors* with corresponding *Architecture type vectors* and consists of 6,934 samples in total. The second is *Keywords vectors* with corresponding *Activation function vectors* and consists of 3,482 samples in total. The third subset is *Keywords vectors* with corresponding *Building blocks vectors* with 2,934 samples in total. At Figure 2 are shown the three datasets with the corresponding number of occurrences of the target variables.

### B. LEARNING THE INTELLIGENT MODELS

The process for learning the intelligent systems consists of three main steps as shown in the "Learning intelligent system" section of Figure 1. The first is referred to as dataset preprocessing since it encompasses additional methods for preparing the dataset to be appropriately used for learning the intelligent models. The second step is applying several methods that are evaluated in the third step and the best models are chosen to be used as intelligent models.

#### 1) Dataset preprocessing

To ensure high-quality data, some preprocessing is needed. The first step is removing samples that do not bring any information. Those are the ones where all the binary indicators for the named entities are zeros. This occurs due to some papers not containing any of the named entities in Table 2 and also the "specialized" NER model might have failed to detect some of them. We separately consider the number of samples for each named entity and disregard named entities below a certain threshold as models struggle to learn meaningful representations internally when samples are few.

During this phase different feature portfolios were used based on the number of selected keywords. This is done in order to find a combination which will capture the most information while also using less number of features. The raw (original) data consisted of 16,123 features corresponding to all distinct keywords. Having a high number of features brings longer model training times and noise. To address this issue, a dimensionality reduction techniques is tested: truncated SVD [35]. Data embedded using truncated SVD was fed into each of the tested intelligent models. Multiple benefits can be observed using embedding techniques such as having data that is easier to work with, faster model training, and more importantly, they allow projection of the data into a lower-dimensional space where semantically similar keywords are closely located to each other. Similarly, deep learning architectures have been proven to achieve good results when paired with embeddings, as shown in [36].

#### 2) Learning the intelligent models

While choosing an optimal ML model, numerous architectures are tested and data manipulations are done to achieve good results. SVM is considered a good option, since according to the research presented in [37], SVM-based methods are more suited for a dataset at which the number of features exceeds the number of instances. Multitude of methods for multi-label learning are compared [37] and results show the binary relevance (BR) [38] method provides good results. Following the advice of the research, the same method is used for the problem at hand. To obtain more reliable results, several other approaches are applied and compared to those obtained from SVM. The architectures used in this research are:

- SVM
- Random forest
- XGboost
- Multilayer perceptron
- Convolutional neural networks

Since the prediction problem belongs to the category of multi-label classification, some of these architectures do not inherently provide multiple output values. To overcome this, two options are available, **problem transformation** and **algorithm adaptation**. Problem transformation converts a multi-label problem into multiple single-label problems, and then trains a separate classifier for each of them. The

final prediction is an aggregated result from each of the independent classifiers. Algorithm adaptation, on the other hand, adapts the original algorithm to work with multi-label data. This is usually done by changing the cost function of the algorithm. SVM is used with the BR method for the purpose to obtain multi-label predictions. Another problem transformation method that was tested and achieved comparable results to BR is label powerset. This considers all the distinct label combinations that appear in the dataset and treats the problem as a multi-class task, meaning a training sample will contain all label combinations, and the original combination of the sample will be one-hot encoded.

To ensure that the distribution of the labels in the training and testing set will be nearly equal, the stratification method has been used to provide a balanced split [39]. Without using stratification, it might be possible for a certain label to appear only in one of the sets and therefore, worsen the generalization performance of the model. Furthermore, the stratification method used iteratively distributes the rarest occurring labels into respective N-folds, since the label distribution can be changed more easily later in the subsequent iterations with the more frequently occurring labels. The size of the datasets before undergoing stratification can be seen in Figure 3.

After stratification the following number of samples were obtained for training and testing the models:

- Architecture type dataset: train: 4,169 / test: 1,900
- Activation function dataset: train: 2,070 / test: 955
- Building blocks dataset: train: 1,737 / test: 784

### 3) Evaluating the intelligent models

The evaluation of the models is conducted using several metrics as follows:

- Precision
- Recall
- F1-Score
- Hamming distance

Because of the nature of the problem (multi-label), some metrics are more suited than others and give reliable information on the models' performance. Most attention is given to F1-score, representing the harmonic mean of precision and recall. It is used to decide if to further experiment with a model's hyperparameters or try different approaches. The basis for choosing the f1-score as a deciding indicator of a model's performance is the equal importance of precision and recall to our problem.

We decided not to use accuracy in any of our evaluations since we experimented with a multi-label problem. Using accuracy, a single prediction must have an exact match between the predicted labels and and the corresponding true labels in order to be classified as a correct prediction. Because of that, it is a harsh metric in multi-label problems, thus other metrics are needed that will take into account even partially correct predictions.

Precision represents the ratio of correctly predicted positive instances to the total predicted positive instances. For the problem at hand, it is used in conjunction with recall, which represents the ratio of positive instances that were retrieved from a certain class. Precision and recall help during training to observe if a model achieves overfitting and testing to see if a specific target variable (label) achieves good results by having representative samples.

Hamming distance represents the distance between two binary vectors. It is calculated as the sum or averaged number of bit differences between the two vectors, making it suitable for our multi-label problem. The averaged number of bit differences is used in this work since having a scaled score between 0 and 1 is easier to interpret.

To take into account the general performance of the models on all target variables, several averaging methods to average each of the metrics except hamming distance are used as follows:

- Micro average
- Macro average
- Weighted average
- Samples average

Micro average computes the metrics by considering the total number of true positives, false positives, and false negatives. Macro average computes the metrics for every label and finds their unweighted mean. This gives equal importance to each label without taking label imbalance into account. The weighted average is similar to the macro average, it computes the metrics, finds their mean, but it also applies weight to each separate label which is the number of true instances for the particular label. This provides results that are heavily influenced by the more frequent labels in the dataset. Finally, samples average works by computing the metrics on each label instance and returning the average.

Due to having unbalanced datasets, some labels will occur more often than others, so it is needed to have a metric that will represent overall model performance on all labels. All of our target variables (labels) are of equal importance, and therefore, while evaluating the models, we give most significance to macro-averaged metrics. This averaging method was also noticed to give the least optimistic values in the majority of cases when compared to the other averaging methods which helps in building a more realistic expectation of the performance of the final model.

Models tend to predict the majority class in the training data and to overcome that tendency, class weight is applied. Using class weights, each incorrect prediction will influence the loss value depending on the assigned weight for that particular class. We use a "balanced" class weight mode during the training phase, which adjusts the weights to be inversely proportional to the class frequencies. This showed better results than without using any class weights. For a representative metric of overall performance, macro averaged f1-score is used, which calculates f1-score for each label and finds their mean. This way, the overall score is penalized if generalization performance is significantly poor on some labels.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3124633, IEEE Access



## IV. RESULTS

Figure 3 presents how many samples each label appears in from each of the three datasets. These samples are obtained after removing the target variables (labels) that occur less than 300 times and also after splitting the dataset into training/testing set where samples that had small keyword overlap in their corresponding keyword vectors were removed. We chose 300 as our threshold number because we wanted to have enough representative samples even after splitting into a train and test set, likewise to decrease the variance in the results. Additionally, most of the less frequent features occurred less than a few hundred times.

In the further processed datasets, it is evident that there is still class imbalance and in order to address it, class weight is applied to every target variable. As a result, while evaluating models, better metrics were achieved on all datasets when using class weight compared to not using it.

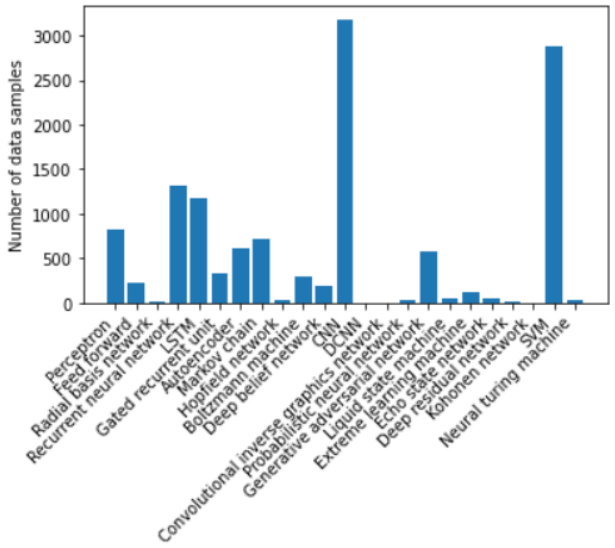
After completing the dataset preprocessing and building the intelligent models for the architecture type, activation function, and building blocks, the best results from all the models are shown in Table 4, which also appear in the same, said order. The numbers shown in the tables indicate the results obtained with using latent semantic analysis (LSA). LSA tries to bring forth latent features that are inherent in the dataset, which in our scenario ended up in 100 latent features. It does this by constructing a 2D-matrix that represents word count per document. After constructing the matrix, it uses singular value decomposition (SVD) to reduce the dimensionality while preserving the similarity between documents.

Looking at F1-score, SVMs offer the best results on the architecture type and activation function dataset, while the best model for the building blocks dataset is the CNN model. This further proves that SVMs work well with tasks that have a larger number of features, confirming the hypothesis proposed in [37].

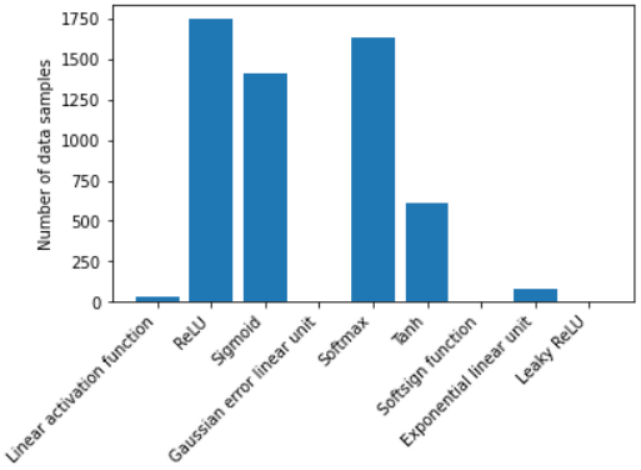
MLPs closely followed the results from the other best models. Surprisingly the best MLP configuration consisted of only one hidden layer with 150 nodes. Deeper MLP architectures were tested, although they were slower to converge and gave slightly worse results. It was noticed that the best MLP results were obtained early in the training process, before the model had managed to significantly converge thus, training was usually stopped after the first 10 or 20 epochs.

Using an ensemble of decision trees, Random Forest (RF) gave the worst f1-score on the activation function dataset, 0.32 and comparable results on the other datasets. SVD is used in order to perform LSA, which in turn will give us lower dimensional data. Principal component analysis (PCA) was also used in the place of SVD and they both achieved similar results. The general poor results on RF are primarily because each decision tree takes a small subset of features, making it difficult to learn as our problem has a large feature space that is shrunk down to a smaller size which is less representative. Also, RF have shown to be prone to overfitting for the problem at hand.

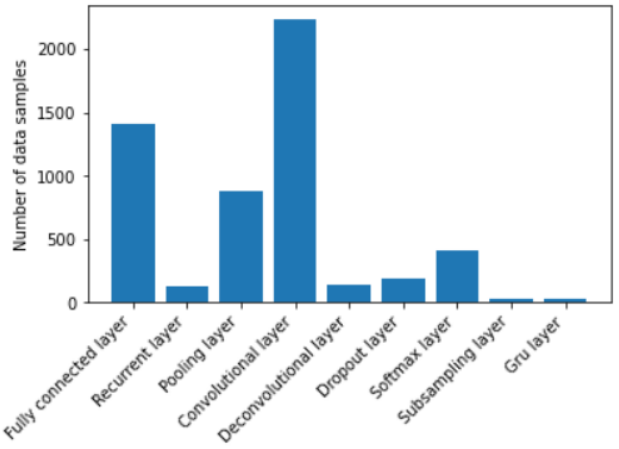
XGboost [40] improves on RF by using gradient boosting.




(a) Number of occurrences for each label in architecture type dataset




(b) Number of occurrences for each label in activation function dataset




(c) Number of occurrences for each label in building blocks dataset

FIGURE 2: Occurrence of labels in different datasets before applying threshold.

| | Precision | Recall | F1-Score | Hamming dist. |
|---|---|---|---|---|
| SVM | 0.43 / 0.50 / 0.46 | 0.58 / 0.57 / 0.53 | 0.46 / 0.53 / 0.46 | 0.2956 / 0.3772 / 0.4540 |
| RF | 0.24 / 0.51 / 0.50 | 0.78 / 0.48 / 0.51 | 0.32 / 0.48 / 0.49 | 0.6720 / 0.3844 / 0.3014 |
| XGboost | 0.57 / 0.50 / 0.45 | 0.37 / 0.44 / 0.43 | 0.43 / 0.46 / 0.43 | 0.1966 / 0.3493 / 0.3333 |
| MLP | 0.71 / 0.49 / 0.52 | 0.31 / 0.44 / 0.40 | 0.40 / 0.46 / 0.43 | 0.1971 / 0.3611 / 0.3046 |
| CNN | 0.52 / 0.45 / 0.48 | 0.38 / 0.57 / 0.54 | 0.43 / 0.50 / 0.51 | 0.4284 / 0.4113 / 0.3262 |

TABLE 4: Results from models (Architecture type dataset / Activation function dataset / Building blocks dataset)

Their sparsity-aware algorithm better handles sparse input but since we embedded our data into a 100 dimensional vector, it doesn't manage to use all of it's benefits. Likewise, gradient boosted trees have also been observed to suffer from quick overfitting.

CNNs are mainly adapted for computer vision, although they can also be used for text classification [41], [42]. The convolutional filters can be trained to extract n-gram features from text [43] and this is one of the reasons for choosing a CNN architecture, since our binary encoded vector can also be viewed as textual data. Inspired by the work of [44], we implemented their nine-layer deep architecture but cut the original number of filters by a factor of 4 due to the limited amount of resources. Their deeper architectures were also tested but all failed to converge.

## V. DISCUSSION

In the process of producing and selecting the best intelligent recommendation models, many data manipulations were done and different architectures were tested. With a high number of keywords, numerous examples have been observed that share a similar meaning between each other, e.g "clustering" and "clusters". This introduces specific noise to the system and embedding provides a way to eliminate it. Another problem of using raw data is the sparsity of the input vectors. Collecting more scientific papers provides more unique keywords that demand more training data for achieving good results. In the end, we are left with a vector which is filled with mostly zeros and just a few features that have a value of one. Further difficulties that arise from the sparse vectors is the small overlap between keywords found in training and test set. After the data underwent stratification, as much as 67.9% of the keywords (from their respectful vectors) found in the train set, did not appear in the test set. This number was obtained from the architecture type dataset, although the other datasets also had small keyword overlap. This represents a significant problem for the generalization performance of the model because during the testing the model encounters keywords it was never trained on. Additionally, the low number of samples per given target label further degrade model performance. We eliminated the small overlap between keywords found in the training and the testing set by leaving only keywords that appear in both sets and removing the rest. Besides removing keywords that do not appear in both train and test set, the training and testing data remain unchanged.

To overcome the low number of samples, pairs of labels are joined together considering their relatedness:

From building blocks:

- Recurrent layer and GRU layer

From Architecture type:

- Perceptron and feed-forward
- Recurrent neural network and gated recurrent unit
- Autoencoder and generative adversarial network
- Markov chain and Boltzman machine

The reason for joining labels is because a positive trend was noticed where increasing the number of samples for a target label also showed better results on all metrics. Furthermore, some labels are similar to each other and are used interchangeably in the literature. Therefore, there is an increase in model performance when comparing the results before and after joining the labels.

Considering dimensionality reduction, while using SVD, the number of components was first set to 2,100 to obtain a very high variance but later was lowered to 100, which is used in general practice. Using 100 components gave better results in contrast to using 2,100 and provided faster training times as well. TSNE was used in data visualization and dimensionality reduction, though it achieved worse results than truncated SVD.

The proposed system can provide some general recommendations that can be used as a starting point to define the architecture for a typical application scenario that further should be tuned concerning its hyperparameters. After that, NAS methods or other AutoML methods should be used to find the best hyperparameters for the specific set of problems. Our approach can be used as a pre-step to define which DNN architecture may be more suitable for a typical scenario since it utilizes the knowledge already published in the state-of-the-art literature. With this in mind, innovation can not be limited by such a system which provides general recommendations.

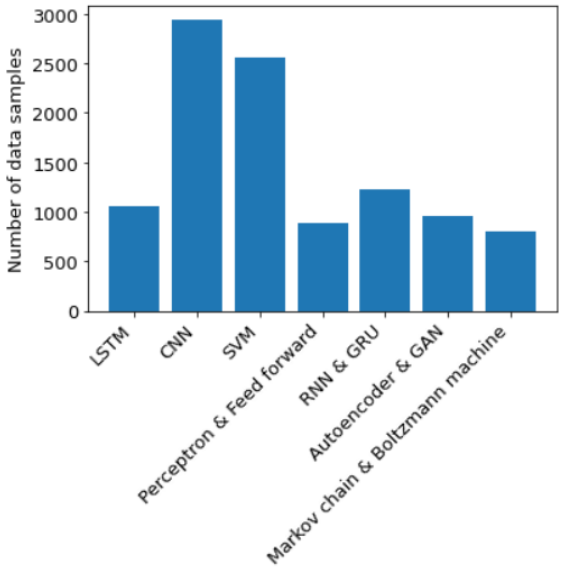
## VI. CONCLUSION

The benefit of our research is multifold, and each is discussed as follows:

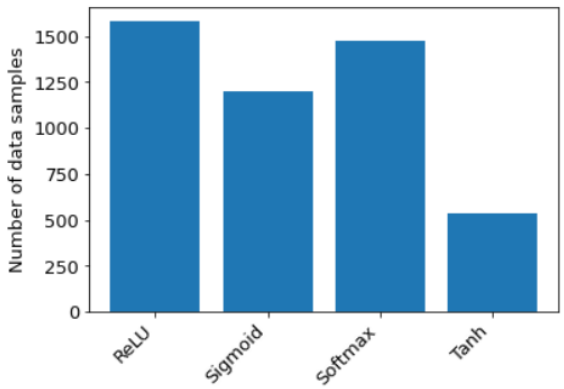
- Establishing a novel database: A novel database was created from a large number of research papers that underlined methods that organized the information in Keyword vectors, Architecture type vectors, Activation function vectors and Building blocks vectors. The database is created to understand how the scientific community builds deep learning methodologies for various problem domains. By processing the corpus of research papers a total of 16,123 unique keywords were

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3124633, IEEE Access

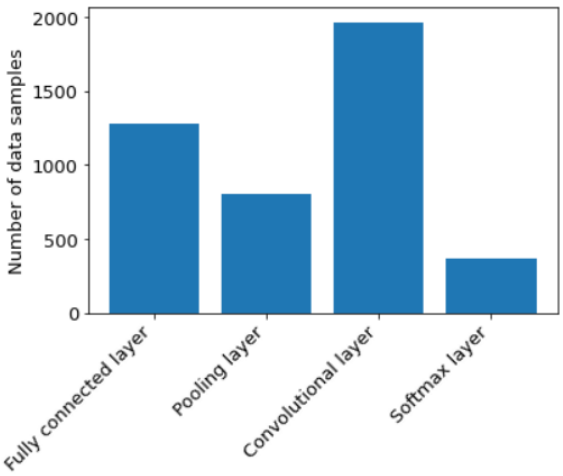





(a) Number of occurrences for each label in architecture type dataset




(b) Number of occurrences for each label in activation function dataset




(c) Number of occurrences for each label in building blocks dataset

FIGURE 3: Occurrence of labels in different datasets after applying threshold.

extracted. Since our proposed database creation process consists of several steps, shown in Figure 1, it allows for independent modifications of its parts, meaning it is possible to create a new recommendation system simply by retraining the NER model to extract named entities from another domain.

- Learning a novel intelligent system: To the best of our knowledge, this is the first time someone has made a data-driven intelligent system for recommending Deep Learning architecture and hyperparameters based on the problem domain represented in form of keywords vectors. We showcase the steps needed to build a database to train such a system and analyze the results gained from an exhaustive dataset manipulation and model building process.
- Contribution to the interdisciplinary research community: A great benefit of such an intelligent system is that it can help the researchers from various fields to build a Deep Learning model that will perform well on their particular problem regardless if they have experience in DL or not. This system will be of a great help to researchers without a computer science background by shortening the gap between them and DL/ML practice.

In the future, we intend to improve our recommendation system by implementing embeddings to our keyword vectors and to give a comparative analysis between using pre-trained embeddings and our own embeddings. Considering the multitude of keywords that have similar meanings, it is even more incentivizing to implement embedding techniques. Next, we are planning to use the outcome of the intelligent system as a stating point to tune it based on the charactestics of the dataset that is being solved. We plan to tackle the problems of certain edge cases that might occur during text processing to obtain higher-quality data. Doing so will improve the quality of the recommendations and make a more robust system.

## FUNDING

This work was supported by the Slovenian Research Agency (research core funding programmes P2-0098 and project Z2-1867).

## REFERENCES

[1] Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. Simple and efficient architecture search for convolutional neural networks, 2017.
[2] Chen Gao, Yunpeng Chen, Si Liu, Zhenxiong Tan, and Shuicheng Yan. Adversarialnas: Adversarial neural architecture search for gans, 2020.
[3] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism, 2016.
[4] Jun Lu, Wei Ma, and Boi Faltings. Compnet: Neural networks growing via the compact network morphism, 2018.
[5] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation, 2017.
[6] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization, 2018.
[7] Minsu Cho, Mohammadreza Soltani, and Chinmay Hegde. Hyperparameter optimization in neural networks via structured sparse recovery, 2020.
[8] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, Dec 2017.

[9] Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data, 2019.
[10] Zhi-Hua Zhou and Ji Feng. Deep forest: Towards an alternative to deep neural networks. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pages 3553–3559, 2017.
[11] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions, 2021.
[12] George Kyriakides and Konstantinos Margaritis. An introduction to neural architecture search for convolutional networks, 2020.
[13] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 415:295–316, Nov 2020.
[14] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications, 2020.
[15] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition, 2018.
[16] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift, 2019.
[17] João Caldeira and Brian Nord. Deeply uncertain: comparing methods of uncertainty quantification in deep learning algorithms. Machine Learning: Science and Technology, 2(1):015002, Dec 2020.
[18] Lara Hoffmann and Clemens Elster. Deep ensembles from a bayesian perspective, 2021.
[19] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016.
[20] Jiefeng Chen, Yixuan Li, Xi Wu, Yingyu Liang, and Somesh Jha. Robust out-of-distribution detection for neural networks, 2020.
[21] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning, 2017.
[22] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc Le, and Alex Kurakin. Large-scale evolution of image classifiers, 2017.
[23] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: Stochastic neural architecture search, 2020.
[24] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware, 2019.
[25] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, pages 5369–5373. International Joint Conferences on Artificial Intelligence Organization, 7 2018.
[26] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer, 2016.
[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.
[28] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.
[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
[30] Ana Gjorgjevikj, Kostadin Mishev, and Dimitar Trajanov. Add: Academic disciplines detector based on wikipedia. IEEE Access, 8:7005–7019, 2020.
[31] F Peng and A McCallum. Accurate information extraction from research papers using conditional random fields. retrieved on april 13, 2013.
[32] Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. Information extraction from scientific articles: a survey. Scientometrics, 117(3):1931–1990, 2018.
[33] Matthew Honnibal and Ines Montani. spaCy: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing, 2020.
[34] Nasi Jofche, Kostadin Mishev, Riste Stojanov, Milos Jovanovik, and Dimitar Trajanov. Pharmke: Knowledge extraction platform for pharmaceutical texts using transfer learning. arXiv preprint arXiv:2102.13139, 2021.
[35] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, 2010.
[36] Kostadin Mishev, Ana Gjorgjevikj, Riste Stojanov, Igor Mishkovski, Irena Vodenska, Ljubomir Chitkushev, and Dimitar Trajanov. Performance evaluation of word and sentence embeddings for finance headlines sentiment analysis. In International Conference on ICT Innovations, pages 161–172. Springer, 2019.
[37] Gjorgji Madjarov, Dragi Kocev, Dejan Gjorgjevikj, and Sašo Džeroski. An extensive experimental comparison of methods for multi-label learning. Pattern recognition, 45(9):3084–3104, 2012.
[38] Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng. Binary relevance for multi-label learning: an overview. Frontiers of Computer Science, 12(2):191–202, 2018.
[39] Konstantinos Sechidis, Grigorios Tsoumakas, and Ioannis Vlahavas. On the stratification of multi-label data. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 145–158. Springer, 2011.
[40] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pages 785–794, 2016.
[41] Ritu Yadav. Light-weighted cnn for text classification, 2020.
[42] Zhenyu Liu, Haiwei Huang, Chaohong Lu, and Shengfei Lyu. Multichannel cnn with attention for text classification, 2020.
[43] Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. Understanding convolutional neural networks for text classification, 2020.
[44] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification, 2017.

GJORGJI NOVESKI obtained his bachelor's degree in computer science from the Faculty of Computer Science and Engineering, Saints Cyril and Methodius University of Skopje, in 2021. Currently he is working at the Jožef Stefan Institute in Ljubljana, Slovenia, as a researcher where he is also pursuing his master's degree. During his studies he has worked on various projects involving 3D object detection, dataset analysis, web platforms and services. As a young and ambitious researcher his current work focuses on machine learning, intelligent systems and natural language processing.

TOME EFTIMOV is a senior reseacher at the Jožef Stefan Institute, Ljubljana, Slovenia. He was a postdoctoral research fellow at the Department of Biomedical Data Science, and the Centre for Population Health Sciences, Stanford University, USA, and a research associate at the University of California, San Francisco, USA (UCSF). He was awarded his PhD degree from the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, in 2018. His main areas of research include statistics, natural language processing, heuristic optimization, machine learning, and representational learning. Specific topics of research include information extraction of food- and nutrition-related concepts from textual data, food data normalization, and knowledge management for food- and nutrition-related data.

KOSTADIN MISHEV is a Ph.D. candidate and a teaching and research assistant in the Faculty of Computer Science and Engineering at Saints Cyril and Methodius University of Skopje. He received a bachelor's degree in informatics and computer engineering, a master's degree in computer networks and e-technologies degree from Saints Cyril and Methodius University, Skopje, in 2013 and 2016, respectively. He started his Ph.D. studies in 2017 in the field of computer science and engineering. In his career, he has participated in numerous software and research projects. As part of his scientific research work, he has published more than 20 scientific papers at international conferences and journals. His research interests include natural language processing, representation learning, speech technologies, semantic web, assistive technologies, web and mobile development.

MONIKA SIMJANOSKA Ph.D., is an assistant professor and senior researcher at the Faculty of Computer Science and Engineering in Skopje, and co-founder and co-owner of research and development company iReason LLC in Skopje, N. Macedonia. She received her Ph.D. degree in the field of Informatics, defending her Ph.D. thesis titled "Bioelectrical and Bioacoustical Signal Processing for Prediction of Medical Conditions".

She is an author and co-author of more than 60 research papers, partly published in journals, on conferences, and as chapters of books. During her research work, she has had multiple research stays at Institute Jozef Stefan in Ljubljana where she worked in the field of Machine learning and biomedical signal analysis. Also, she had short research stays at University College Dublin, Dublin Ireland; Maison Jean Kuntzmann, Université Grenoble Alpes, Grenoble, France, and Laboratory for Microelectronics at the Faculty of Electrical Engineering, University of Ljubljana, Slovenia.

She received an award for best paper and FESTO Young Researchers and Scientists Support Scholarship for the proposed platform architecture for colorectal cancer analysis at the 25th DAAAM International Symposium, Vienna, Austria. Being trustworthy among the data science community is the leading factor for her dynamic and diverse career development encompassing intelligent systems creation as a response by the various requirements of the society. She has had arrangements in biomedicine, bioinformatics, natural language processing, social sciences, agriculture, cloud computing, education and recently in microprocessor systems.