When to Use Which Neural Network?
Finding the Right Neural Network Architecture for a Research Problem

Michael Färber[1] **and Nicolas Weber**[2]

[1] Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
[2] Heidelberg University, Heidelberg, Germany
michael.faerber@kit.edu

Abstract

Considering the increasing rate of scientific papers published in recent years, for researchers throughout all disciplines it has become a challenge to keep track of which latest scientific methods are suitable for which applications. In particular, an unmanageable amount of neural network architectures has been published. In this paper, we propose the task of recommending neural network architectures based on textual problem descriptions. We frame the recommendation as a text classification task and develop appropriate text classification models for this task. In experiments based on three data sets, we find that an SVM classifier outperforms a more complex model based on BERT. Overall, we give evidence that neural network architecture recommendation is a nontrivial but gainful research topic.

1 Introduction

A multitude of neural network architectures has been proposed, with many more to come. The knowledge graph Wikidata,[1] for instance, models 66 variants of neural network architectures. Machine learning researchers and practitioners, such as data scientists and software developers, are increasingly confronted with the question: When to use which neural network architecture?[2]

So far, approaches to neural architecture search and search engines for research data management have been proposed. Neural architecture search (Elsken, Metzen, and Hutter 2019) is concerned with the task of automatically finding the optimal neural network architecture design for a specific task. However, neural architecture search approaches usually restrict themselves to a specific architecture type (e.g., RNN or CNN) and target finding the optimal architecture, such as the number of network layers or hyperparameters. Instead, the focus of this paper is on a different level of granularity. The idea is to create a model that finds the most suitable neural network architecture for a research problem described in natural language. Furthermore, neural network search engines and ontologies, such as FAIRnets (Nguyen and Weller 2019; Nguyen et al. 2020), differ from us because they allow only keyword queries. Chen et al. (2019) found that real information needs are most often formulated as phrases and not as keywords. The latter case constitutes only 32% of the investigated queries. In addition, such search systems retrieve specific neural network instances instead of neural network architectures.

In this paper, we propose the task of *neural network architecture recommendation*. It differs from other domain-specific text classification tasks in the fact that research problem descriptions as input are largely not available and first need to be created. To this end, we propose two methods that extract the problem descriptions from papers' abstracts. In addition, the usage of neural network architectures is highly imbalanced in the literature, making the recommendation task a nontrivial challenge. We train and evaluate two state-of-the-art machine learning-based approaches for neural network architecture recommendation, using the extracted research problem descriptions and neural network architectures derived from Wikidata. Our proposed approach can benefit students as well as researchers of various domains. For researchers with little expertise in the field of machine learning in particular, our approach simplifies the process of selecting a suitable neural network model and presumably yields a reduction in time spent on preliminary research on appropriate neural architectures.

To summarize, we make the following contributions:

1. We create evaluation data sets for neural network architecture recommendation, consisting of 66 unique architectures and 284,337 textual problem descriptions.
2. We train and evaluate several classifiers capable of predicting neural network architectures based on textual problem descriptions.[3]

The paper is structured as follows: In Section 2, we describe the creation of the neural network architecture set, as well as two data sets with scientific problem descriptions. Section 3 discusses the methods to predict the neural network architectures based on textual descriptions. In Section 4, we present our experiments. We conclude in Section 5 with a summary.

[1]See https://www.wikidata.org/.

[2]See https://datascience.stackexchange.com/questions/20222/how-to-decide-neural-network-architecture.

[3]All data and source code is available online at https://github.com/michaelfaerber/NNARec.

Kernel Perceptron	RNTN	RoBERTa	GCN
Multilayer Perceptron	TDNN	Neocognitron	CNN
Restricted Boltzmann Machine	Bcpnn	Cresceptron	GRU
winner-take-all	MCDNN	Modular NN	SNN
Hopfield N	HONN	Deep NN	DNC
Neural Abstraction Pyramid	Elman N	Feedforward NN	PNN
Shift Invariant NN	RecCC	perceptron	DBN
Spatial Transformer Network	Jordan N	Highway N	GAN
Neural History Compressor	ADALINE	Transformer	ESN
Kohonen NN	LSTM	AlexNet	ELM
Radial Basis Function N	CPPN	Text-CNN	VAE
Connectionist Expert System	CMAC	EntNet	HTM
Boltzmann machine	PCNN	Hamming NN	LSM
Bidirectional Associative Memory	DRPNN	LeNet-5	RNN
Neural Turing Machine	NNPDA	Stochastic NN	DQN
Self-organizing Map	MANN	CapsNet	
ResNet	RecNN	3D-CNN	

Table 1: Neural network architectures in Wikidata.

2 Data

2.1 Set of Neural Network Architectures

Our approach utilizes the knowledge graph Wikidata to obtain a list of neural network architectures. The following aspects are taken into consideration: (1) all subclasses of artificial neural networks; (2) the hierarchical structure of these subclasses; (3) aliases and abbreviations. Our query returns 67 results, of which 66 (see Table 1) are appropriate for the task at hand (the additional item returned is the "artificial neural network" item itself).

2.2 Problem Descriptions

Our aim is to recommend neural network architectures based on problem descriptions. However, problem descriptions are, to the best of our knowledge, not available to a large degree. However, we argue that parts of papers' abstracts are a good approximation of textual research problems. Thus, we use the paper abstracts and metadata from the Microsoft Academic Graph (MAG; Sinha, Shen et al. (2015)).

We only consider English abstracts in which neural network architecture names are mentioned. After carefully analyzing the resulting abstracts, an issue related to the neural network architecture "transformer" is found. Because the word "transformer" is polysemic, the bulk of abstracts mentioning transformers are mostly concerned with (electrical) engineering. To circumvent this problem, these abstracts are filtered by a keyword list.[4] After this, 284,337 abstracts remain.

The abstracts usually include both problem descriptions and names of associated neural network architectures. To extract these items, we propose the following methods.

Extraction by Abstract Splitting. The first approach of creating a data set is based on the observation of Jiang et al. (2012). The main idea is that abstracts can often be conceptually split into an introduction and a solution part. After manually checking 500 randomly selected papers from four conferences (SIGIR, SIGKDD, RecSys, and CIKM), the result indicates that 71% of the abstracts adhere to this structure (Jiang et al. 2012).

[4][BERT, GPT-2, GPT-3, natural language, self-attention]

Example Problem Description: The prediction of failures in rotating machines is an important issue in industries to improve safety, to reduce the cost of maintenance and to prevent accidents.
Example Solution: In this paper a predictive maintenance algorithm, based on the analysis of the orbits shape of the rotor shaft is proposed. It is based on an autonomous image pattern recognition algorithm, implemented by using a **Convolutional Neural Network (CNN)**.[...]
Example Target Label: CNN

Table 2: Example of *extraction by abstract splitting* (abstract from Caponetto et al. (2019)).

We observe that the key phrases "in this paper" and "this paper" play an important role in the transition between the problem statement and solution parts (see Table 2). We therefore check for each sentence in the abstracts whether these key phrases occur. If there is a match, we mark the sentence as the beginning of the solution part and all prior sentences as the problem description part. Table 2 provides an illustration of our abstract-splitting approach.

To evaluate the effectiveness of this method, we let two experienced researchers classify 500 randomly selected splits into the following categories: (1) the split is correct, (2) the split is incorrect, but a correct split is possible, and (3) the abstract cannot be split into an introduction part and a solution part. The differences between the annotators lie mostly in the annotators' conceptions of where to set a split, rather than whether a split is possible. Inter annotator agreement can be reported by Cohen's kappa of 0.7538, which indicates a good agreement for this task. Overall, based on our analysis, 88.6% of the randomly sampled splits are evaluated as being correct.

Once the abstracts have been split, only parts of the abstracts with mentions of neural network architectures in their respective solution part are included in the data set, with the introduction parts as problem descriptions and the neural network architectures as the labels. We will refer to the resulting data set as the **Abstract Splitting (AS)** data set.

Extraction by Key Phrase Templates. The aforementioned method has the drawback that the neural network mentioned in the solution part of an abstract is directly related to the problem description outlined in the first part of this abstract. However, problem descriptions in other parts of the abstract are ignored. To combat this issue, we create a method of identifying problem descriptions more precisely.

In a first step, we analyze the abstracts that contain neural network architecture mentions to obtain an understanding of recurring phrases in problem descriptions. From these phrases, we then create templates to extract problem descriptions in all abstracts. Table 3 illustrates an example of a template and a match. Overall, we came up with 44 templates that are based on regular expressions.

As we can see in Table 3, this method generally results in shorter problem descriptions than the plain abstract splitting method proposed above. As only the problem descriptions

Template	we use(d) <METHOD> for/to <PROBLEM>
Matches	Specifically, we use a simple yet powerful architecture, consisting of only one CNN and a single resolution input, combined with a new loss function for pixel-wise fixation prediction during free viewing of natural scenes.

Table 3: Example for problem extraction by keyphrase templates.

and the neural network architecture names are of interest and not the long method descriptions, we additionally identify the neural network architecture names mentioned in METHOD (in the example in Table 3: CNN) given our list of neural network architecture names. To minimize redundancy for extractions made in the same abstract, if one string is a substring of the other, the longer one is chosen and the other one is dismissed.

A last step to reduce noise is to filter out common phrases in the texts that carry no information (e.g., "solve this problem" given the template "we use METHOD to solve this PROBLEM"). While the quality of the extracted problem descriptions is overall satisfying, from 284,337 abstracts mentioning neural network architectures, only 35,829 problem descriptions remain based on this method. The resulting data set is designated the **Key Phrase Extraction (KE)** data set.

2.3 Neural Network Architecture Mentions

Due to the differences in the data set creation, the distribution of neural network architectures differs in our AS and KE data sets. To make them comparable, we take two steps. First, to avoid losing all instances of sparse classes, the hierarchical structure of some neural network architectures allows for the inclusion of some sparse classes into their parent classes (e.g., GRU is integrated into RNN). We perform this step for all classes with less than 200 instances, given there is a hierarchy to exploit. Second, because some architectures are rarely mentioned, only classes with at least 200 instances in both data sets are considered. This leads to both data sets containing the same classes. From the initial 66 neural network architectures retrieved from Wikidata, only 15, which are listed in Figure 1, remain.

2.4 Preparing AGENDA as Test Set

The *Abstract GENeration data set* (AGENDA; Koncel-Kedziorski et al. (2019)) has been used for automatic text generation based on knowledge graphs and consists of knowledge graphs paired with paper titles and paper abstracts from the AI domain. As mentions of tasks and methods are also labeled in these paper abstracts, we can use this data set for an additional, complementary evaluation, particularly as an additional test data set considering its size.

It is important to note that the text spans labeled as problem descriptions in this data set are rather short to be more compatible with knowledge graph entities.

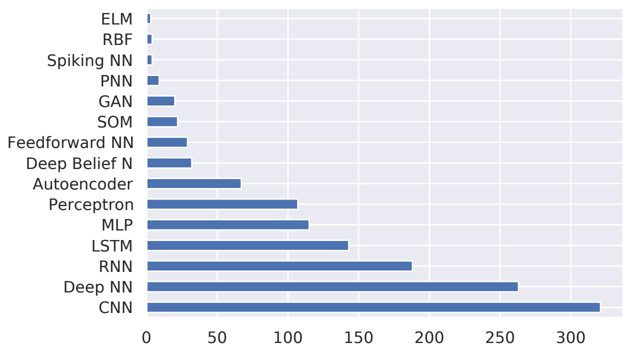

Figure 1: Neural network architecture counts in the modified AGENDA data set.

We therefore increased the context by considering whole sentences as problem descriptions. The resulting, modified data set, designated **mod-AGENDA**, has 1,327 instances, distributed over 15 classes, as Figure 1 shows.

3 Methods

The task in this paper falls into the realm of supervised classification. The overwhelming majority of instances in each of our our data sets has only a single label. Thus, in the following evaluation, we consider the task as a multiclass, single-label classification task. For this paper, we consider the following widely used text classification approaches.

TF-IDF + SVM. One approach is based on SVM, using TF-IDF for representing the text as vectors. As this can lead to very high dimensional sparse vectors, it makes sense to filter out stopwords for the vector representation.

BERT + Classification Layer. As our second approach, we use a fine-tuned BERT-model with an additional classification layer.

4 Experiments

4.1 Evaluation Settings

We use a train-test split of 80:20 for the AS and KE data sets. Each of the methods is trained and tested on either the AS data set or the KE data set. In addition, the models trained on the KE and AS data sets are evaluated on the modified AGENDA data set to evaluate the generalizability of the approaches.

We considered the following methods: (1) *SVM.* We used scikit-learn's TfidfVectorizer for numeric representations and an SVM implemented via a one-vs-rest classification scheme. (2) *Fine-tuned SciBERT.* We use SciBERT (Beltagy, Lo, and Cohan 2019), a scientific domain-specific, pretrained BERT-model, and fine-tune it on the classification task with Adam optimizer (Kingma and Ba 2014). (3) *Most frequent class (MFC).* We consider the MFC as a baseline.

Training Data	Test Data	Method	Precision (Macro)	Recall (Macro)	F1 (Macro)	Accuracy
KE	KE	MFC	0.0246	0.0667	0.0359	0.3688
KE	KE	SVM	**0.5280**	**0.4242**	**0.4629**	**0.5908**
KE	KE	SciBERT	0.2198	0.2404	0.1793	0.2576
AS	AS	MFC	0.0219	0.0667	0.0330	0.3284
AS	AS	SVM	**0.5973**	0.3893	**0.4355**	**0.5711**
AS	AS	SciBERT	0.3423	**0.4178**	0.3391	0.4009
–	mod-AGENDA	MFC	0.0034	0.0667	0.0064	0.0505
KE	mod-AGENDA	SVM	**0.1304**	**0.1030**	0.0694	0.0980
KE	mod-AGENDA	SciBERT	0.0812	0.0757	0.0481	0.0663
AS	mod-AGENDA	SVM	0.1186	0.0880	**0.0755**	**0.1017**
AS	mod-AGENDA	SciBERT	0.0569	0.0850	0.0576	0.0950

Table 4: Results of most frequenct class (MFC), SVM, and fine-tuned SciBERT.

4.2 Evaluation Results

Precision, recall, F1-score[5] (all macro-averaged), and accuracy for the MFC baseline, SVM, and fine-tuned SciBERT are reported in Table 4. The results show that the SVM classifier trained and tested on the KE data set is most successful with respect to recall, F1 score, and accuracy. It beats the more complex SciBERT classifier by more than 100 % in accuracy (0.5908 vs 0.2576) and F1-score (0.4629 vs 0.1793). However, we note that accuracy is not an excellent metric for unbalanced data sets.

Regarding the classifiers trained and tested on the AS data set, the SVM also beats the SciBERT model with respect to precision, F1 score, and accuracy, but with less significance. Here, the accuracy of the SVM is 0.17, and the F1-score is 0.1 higher than that of SciBERT.

SVM and SciBERT trained on the AS and KE data sets perform superior in most cases compared to the MFC baseline. Notably, MFC achieves a higher accuracy than SciBERT on the KE data set.

When evaluating the approaches on the mod-AGENDA data, the results drop significantly. Nonetheless, the SVM classifier still achieves the best results, with only little difference between the AS and KE data sets as training data sets. SciBERT still outperforms the MFC baseline.

The methods trained on the AS data set generalize better to some degree than the methods trained on the KE data set, despite the simple creation process of the AS data set. A likely reason for this phenomenon is that the AS data set is more similar to the AGENDA data set than the KE data set. In particular, the research problem descriptions in the KE data set are much shorter than in the AS data set.

Overall, given 0.59 and 0.57 as the best accuracy scores and 0.46 and 0.44 as the top F1 scores, we come to the conclusion that neural network recommendation based on textual task descriptions is a nontrivial task (motivating our paper), while it indicates that users (e.g., early-career researchers) might find such recommender systems helpful.

[5]The F1-score is calculated as the arithmetic mean over the individual F1 scores (Opitz and Burst 2019).

5 Conclusion

This paper introduced the task of recommending neural network architectures based on textual problem descriptions. To this end, we created two data sets of labeled problem descriptions. The first splits abstracts by means of signaling phrases and labels the problem parts by matching neural network-architecture names. The second method uses recurring phrases to extract shorter and more precise problem descriptions via regular expressions. We used both data sets to train and evaluate classifiers. We identified the SVM-based approach as a promising method, outperforming a BERT-based approach.

In the future, we will extend our recommender system to machine learning methods in general and combine it with the recommendation of other scholarly entities, such as data sets (Färber and Leisinger 2021). Furthermore, we plan to provide a running system for neural network architecture recommendation accompanied with a user study.

References

Beltagy, I.; Lo, K.; and Cohan, A. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.

Caponetto, R.; Rizzo, F.; Russotti, L.; and Xibilia, M. 2019. Deep lLarning Algorithm for Predictive Maintenance of Rotating Machines Through the Analysis of the Orbits Shape of the Rotor Shaft. In *Proceedings of the 1st International Conference on Smart Innovation, Ergonomics and Applied Human Factors*, SEAHF'19, 245–250.

Chen, J.; et al. 2019. Towards More Usable Dataset Search: From Query Characterization to Snippet Generation. In *Proceedings of 28th ACM International Conference on Information and Knowledge Management*, CIKM'19, 2445–2448.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural Architecture Search: A Survey. *J. Mach. Learn. Res.*, 20: 55:1–55:21.

Färber, M.; and Leisinger, A. 2021. Recommending Datasets for Scientific Problem Descriptions. In *Proceedings of the The 30th ACM International Conference* *on Information and Knowledge Management*, CIKM'21, 3014–3018.

Jiang, Y.; Jia, A.; Feng, Y.; and Zhao, D. 2012. Recommending Academic Papers via Users' Reading Purposes. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys'12, 241–244.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koncel-Kedziorski, R.; Bekal, D.; Luan, Y.; Lapata, M.; and Hajishirzi, H. 2019. Text generation from knowledge graphs with graph transformers. *arXiv preprint arXiv:1904.02342*.

Nguyen, A.; and Weller, T. 2019. FAIRnets Search - A Prototype Search Service to Find Neural Networks. In *Proceedings of the International Conference on Semantic Systems*, SEMANTiCS'19.

Nguyen, A.; Weller, T.; Färber, M.; and Sure-Vetter, Y. 2020. Making Neural Networks FAIR. In *Proceedings of the Second Iberoamerican Conference and First Indo-American Conference*, volume 1232 of *KGSWC'20*, 29–44. Springer.

Opitz, J.; and Burst, S. 2019. Macro F1 and Macro F1. *CoRR*, abs/1911.03347.

Sinha, A.; Shen, Z.; et al. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *Proceedings of the 24th International Conference on World Wide Web Companion*, WWW'15, 243–246.