Optimizing Neural Networks for Patent Classification

Louay Abdelgawad[1]✉, Peter Kluegl[1], Erdan Genc[1], Stefan Falkner[2], and Frank Hutter[2]

[1] Averbis GmbH, Freiburg, Germany
first.last@averbis.com
[2] Machine Learning Institute, Albert-Ludwigs University of Freiburg, Germany
sfalkner@informatik.uni-freiburg.de
fh@cs.uni-freiburg.de

Abstract. A great number of patents is filed everyday to the patent offices worldwide. Each of these patents has to be labeled by domain experts with one or many of thousands of categories. This process is not only extremely expensive but also overwhelming for the experts, due to the considerable increase of filed patents over the years and the increasing complexity of the hierarchical categorization structure. Therefore, it is critical to automate the manual classification process using a classification model. In this paper, the automation of the task is carried out based on recent advances in deep learning for NLP and compared to customized approaches. Moreover, an extensive optimization analysis grants insights about hyperparameter importance. Our optimized convolutional neural network achieves a new state-of-the-art performance of 55.02% accuracy on the public `Wipo-Alpha` dataset.

Keywords: Text classification · Deep learning · Patent classification · Hyperparameter optimization

1 Introduction

A patent is a document issued by a governmental office in order to protect the rights of an invention from being produced, used, or sold, without the permission of the inventor. For an invention to be patented, it has to fulfill three characteristics; it has to be novel, it has to provide an improvement step of something already available, and it has to be implementable by the industry.[1]

According to the Patent Technology Monitoring Team[2], the number of patents filed every year has increased from $417,508$ to $629,647$ between the years 2005 to 2015. It is important to assign each patent to its corresponding category in order to be later reviewed by the suitable domain examiner, who can decide

[1] `https://www.epo.org/applying/basics.html` (accessed June 20, 2019)
[2] `https://www.uspto.gov/web/offices/ac/ido/oeip/taf/us_stat.htm` (accessed June 20, 2019)

whether the patent should be granted or not. However, automating this process is complicated for different reasons.

As the focus of research changes over time, the patent categories change as well. Some categories are merged into one category, new categories are added, or even the definition of a category changes [11]. Furthermore, the label distribution is highly imbalanced as patents tend to follow a Pareto-like distribution; i.e. 80% of the patents fall into 20% of the categories [1, 9].

Another issue is the distribution of words in the patents [11]. Unique words may be decisive in the patent classification process, but due to their rarity they are often discarded at the preprocessing step. Moreover, patents usually contain many scientific terms and to avoid plagiarism or to obfuscate relatedness the statements are often described in a complicated manner. In conclusion, patents are manifold, well-structured and long documents. The patent classification categories are large, complex, time-variant, and non-uniformly distributed hierarchies. Therefore, unlike other types of content which allows an easier categorization, patent classification is an ambitious and challenging task.

There exist several publications about automatic hierarchical classification of patents, mainly based on classical machine learning approaches like SVMs. In combination with elaborate feature engineering they achieved state-of-the-art results [11].

In recent years, the NLP community experienced major improvements in text classification driven by the rise of neural network models like recurrent neural nets (RNN) and convolutional neural nets (CNN). Yet, these models are mainly applied on short text passages, e.g. sentiment analysis or question type classification, not on large documents like patents [17].

This work compares recent approaches for text classification applied on the patent classification task. We investigate the boundaries on how much a neural network can be improved with a variety of different word embeddings and hyperparameter optimization. A detailed report of our optimization findings is provided to guide other researchers.

The experimental results are based on two different datasets, a non-disclosed collection of patents and a freely available dataset for comparison to previous approaches and reproducibility [8]. Our optimized convolutional neural network model achieves a new state-of-the-art performance improvement from 49.02% [24] to 55.02% accuracy on the freely available dataset. Likewise, the CNN model outperforms our baseline, a hierarchical SVM (cf. 3.4), from 58.72% to 65.43% accuracy.

The rest of the paper is structured as follows. Section 2 provides related work for patent classification and text classification based on neural networks. Section 3 describes the general task and the approach applied in this work. An empirical study that illustrates our results is presented in Section 4. Section 5 concludes with a summary.

2 Related Work

Text classification, or document classification more specifically, are typical Natural Language Processing (NLP) tasks which deal with the automated assignment of one or multiple pre-defined labels to a given text or document respectively. Over the years, document classification has been applied to many different areas to overcome error-prone and cumbersome manual labeling. Patent classification is the task of assigning hierarchical single or multi-labels to patent documents.

There exist different classification hierarchies depending on the patent office in which the patent is filed [11]. This work focuses on classification based on International Patent Classification (IPC) and the Cooperative Patent Classification (CPC) hierarchies, which are further described in Section 3.1.

The task has already been discussed and evaluated with most approved document classification approaches in several publications. Gomez et al. [11] conducted a comprehensive survey which highlights the challenges of the task and summarizes previous results. In the following, the focus will be shifted towards more recent advances in the field, which are driven by the advent of deep neural nets. In the scope of this work, a hierarchical Support Vector Machine, a customized approach that has been performing very well on the task [10], will serve as a baseline (cf. 3.4).

Early improvements in text classification with deep learning started with the Dynamic Convolutional Neural Networks (DCNN) method [17]. The authors first adopted Convolutional Neural Networks (CNNs), a model well-received in the computer vision domain, to the field of NLP. Following this work, Yoon Kim created another CNN architecture [18]. The main improvement was to embed the input words using pre-trained word embeddings [21] before passing them into the neural network. Moreover, Kim's CNN, as the author named it, uses a single stage of wide parallel convolutions instead of several stacked convolutions on top of each other.

Zhang created a network which works on character-level instead of word-level [30]. Therefore, it does not require any pre-knowledge of the words. Following the success of very deep networks in the field of computer vision such as ResNet [26] and DenseNet [14], one character is read at a time and then fed to convolution and max-pooling layers. Shortly after, Conneau et al. suggested a network to further increase the depth [5] by introducing shortcut links, another idea introduced in ResNet [26].

Many researchers discovered that the combination of CNN and RNN architecture is beneficial, leading to better results. This is due to the different advantages of CNNs and RNNs [27]. The convolutional network has the advantage of extracting higher-level features that are invariant to local translation. On the other hand, a recurrent network is able to capture long-term dependencies. Xiao et al. created an architecture named ConvRec [27], which extends Kim's CNN with a BiLSTM between the convolutional layers and the classification layer.

Kowsari et al. introduced an architecture, consisting of two networks, which is able to use hierarchical features of labels [19]. One network predicts the first

level of the label and passes it on to the second network which predicts the full label.

Yang et al. applied attention mechanisms on document classification [28]. The idea is that the contribution or importance of word in a text can vary based on the context. As a consequence, when classifying text, not all words should be weighted the same. Therefore, an attention layer is built on top of a bi-directional gated recurrent unit (GRU) [3] in order to highlight words at positions with high impact on the label.

Howard et al. created ULMFiT [13], which employs transfer learning. Instead of using word embeddings, a language model is used to represent words. Afterwards, the model is finetuned and the output layers are adjusted to fit the need of the classification task.

Recent trends employing pre-trained language models and context awareness include ELMo [23] and BERT [6]. Bidirectional Encoder Representations from Transformers (BERT) achieves state-of-the-art performance on eleven natural language processing tasks by adding an output layer on top of the pre-trained language model. The language model is trained by masking words in sentences and learning to predict these masked words from their context.

There are only few publications discussing patent classification using deep learning especially with freely available methods or datasets for comparison. Risch et al. solve the same task using an RNN and self-trained word embeddings in RNN-patent [24]. Likewise, Grawe et al. use LSTM with Word2Vec embeddings to classify patents [12]. However, in their work they only consider 50 different categories.

3 Patent Classification

3.1 Patents and Codes

Patents can be understood as rights granted to inventors that allow them to take legal actions against anyone using their invention without permission. According to Gomez et al. [11], patents are usually organized in the following structure:

- **Title:** Patent's name.
- **Bibliographical data:** Contains the ID number of the patent, the names of the inventor and the applicant, and the citations to other patents and documents.
- **Abstract:** A brief overview or description of the patent.
- **Description:** A more in-depth explanation of the invention.
- **Claims:** Legal distinction of the patent and its application fields.

A patent classification system is a hierarchical system used for categorizing patents into different classes, in which patents discussing similar topics are grouped together under the same label.

There exist several patent classification schemes that patent offices abide by. Two popular schemes are the International Patent Classification scheme (IPC)

and the Cooperative Patent Classification scheme (CPC) which are used in this paper. The classification hierarchy is divided into sections, then classes, subclasses, main groups and finally subgroups. Table 1 shows how the hierarchical structure of the G06K9/6296 label is broken down.

One patent may have more than one label assigned to it, as it can cover different topics at the same time. However, in the scope of this work, the classification is treated as single label task, i.e. only the most important CPC code is considered.

Table 1. Breakdown of the G06K9/6296 label using the CPC scheme.

Structure	**Label**	**Description**
Section	G	Physics
Class	G06	Computing; Calculating; Counting
Subclass	G06K	Recognition of data; Presentation of data; Record careers
Main group	G06K9/62	Methods or arrangements for recognition using electronic means
Subgroup	G06K9/6296	Graphical models, e.g. Bayesian networks

3.2 Preprocessing

This subsection describes the applied preprocessing steps for all the experiments. Firstly, the patent title, abstract, description, and claims are concatenated together into a single text. As the types of CNNs used in this work require a constant sized input length, the texts are truncated to 1500 words. This maximum sequence length was identified empirically in previous evaluations. Shorter sequences are padded using white spaces. Furthermore, the text is normalized by lower-casing, removing all non-alphabetic characters and reducing all multi-spaces to a single white space. To split the text into words, the default Keras tokenizer[3] is used. Finally, only the $20,000$ most frequent words are considered.

3.3 Word Embeddings

A word embedding describes a mapping that translates single words or phrases taken from a vocabulary into an n-dimensional real-valued vector space. It is usually the rationale to find a representation which preserves syntactic and semantic attributes and can be passed on to a machine learning algorithm.

The choice of an effective word embedding depends on a large variety of parameters, e.g. the model itself, embedding size, the corpora used for training or the action taken for unknown words. Many standardized and well-described

[3] https://keras.io/preprocessing/text/ (accessed June 20, 2019)

word embeddings, trained on different corpora, can be found online. Among these, the three most popular models (GloVe [22], Word2Vec [21] and FastText [2]) are tested based on different embedding sizes.

3.4 Applied Methods

In the experiments of this work, we compare several approaches ranging from classical SVM to very recently published neural networks. Two methods are described in more detail in the following. We apply a hierarchical SVM model as a strong baseline and Kim's CNN as a neural network for further optimization.

Hierarchical SVM These models proved to be successful in production environments. They provide a good trade-off between training speed, prediction speed, accuracy, and memory footprint. The hierarchical structure, in form of a tree, is aligned to the CPC scheme. Each node of the tree consists of a single-label classifier implemented by several one-vs-all linear SVMs. In order to avoid some ambiguities in the CPC codes and to improve the accuracy, the first two levels have been merged, i.e. the first node in the hierarchical model classifies labels like G06 (cf. Table 1). The final prediction of the model is determined using a greedy search on the product of the softmax confidences of each node in the tree path. The features of the single SVMs consist of a bag-of-stems of the first 2000 characters of each section. The features are weighted using logarithmic frequencies with redundancy [20] and L2-normalized. Overall, hSVM scales considerably better in terms of amount of labels compared to non-hierarchical SVMs and is therefore applied especially for the large dataset (cf. 4.1).

Yoon Kim's CNN A recap of Kim's CNN is presented in this section as it is the most used method in this work. The preprocessed text is fed to Kim's CNN through an embedding lookup, which converts word IDs to vectors represented in a highdimensional vector space. Afterwards, one or more 1-D convolutional layers are applied on top of the embedding layer. These convolutional layers may have different sizes and the convolutional filters are typically named regions. Furthermore, each of these regions have filters with different weights. A max-over-time pooling is then applied to the output of the convolutional layers. Thereafter, the output of all the layers is concatenated and fed to a softmax in order to obtain a probability distribution over the label classes.

3.5 Optimization and Assumptions

Hyperparameter Optimization Neural network parameters can be divided into two types, the normal and the hyperparameters. Normal parameters, such as weights and biases, are changed during training. Hyperparameters, such as learning rate and batch size, are set before the training begins. Hyperparameter optimization describes the process of tuning these parameters by running

different configurations in order to choose the optimal one. Hyperparameter optimization techniques range from simple random search to more sophisticated, efficient methods such as Bayesian Optimization (BO). In this work, BOHB, a state-of-the-art hyperparameter optimization technique by Falkner et al. [7], is used to tune the parameters. It combines the best of two worlds leveraging the strong performance of BO while maintaining the speed of hyperband (HB).

Assumptions Three main assumptions are set up to limit the computing time to a feasible maximum when comparing different models with different parameter configurations.

First, all default hyperparameters are comparable, i.e. if model x is better than model y based on the default settings, x will remain better than y, when tuning the parameters of both models.

Second, if model x performs better than model y on a subset, then x will remain better than y on the whole dataset.

Third, improvements made on a subset have the same effect on the whole dataset, i.e., if for example a parameter is tuned on a subset, then this change will reflect on the whole dataset as well.

We note that in practice these assumptions only hold approximately, but they do motivate the choices made in our algorithm.

4 Experimental Results

4.1 Datasets

Two collections of datasets are utilized in this work. The first dataset called `Pat` consists of a non-disclosed collection of 1.05 million English patents that have been filed to a patent office. The patents contain complete sections like title, abstract, claims and description with 2500 words on average and are labeled according to the CPC scheme. This dataset represents the main task of routing new patent applications to the correct expert examiner. Each patent is assigned to a range of CPC codes, grouped by the starting code of the range. These ranges represent the area of expertise of the examiner groups. Considering these ranges as target labels leads to a single-label classification task with an overall amount of 1382 classes. Due to its real world nature the dataset is highly imbalanced. The most frequent class includes around 30100 instances, while around 30% of the classes include less than 100. The average number of documents per class on each CPC level decreases considerably from $151,296$ examples on the 1^{st} level to 31 examples on the 5^{th} level. For testing the models on this dataset, a separate set of $118,177$ patents with the same class-distribution is utilized.

A subset of `Pat` called `Pat16` contains $250,017$ patents filed in 2016 and applies a reduced set of 8 classes corresponding to the first level of the CPC code. This reduced and simplified dataset is used in different variations of optimization for performance reasons. As a separate test set for `Pat16`, $115,795$ patents filed in 2017 are utilized.

For comparable results, a second, freely available, dataset is used. Among other `Wipo` datasets [4] and the CLEF-IP dataset [5], `Wipo-Alpha` was chosen as it provides text for all sections and a distinguished single CPC/IPC class for the single-label classification.

`Wipo-Alpha` contains $75K$ English patents with overall 451 classes. The separate test set consists of 28,926 patents. It was used in previous evaluations and shares most characteristics with the main dataset.

4.2 Model Selection

Several previously published models with good results are evaluated on both datasets `Pat16` and `Wipo-Alpha`, in order to pick a reasonable model for further optimization. The default hyperparameters mentioned in the publications of each model are used for the preliminary comparison, except for BERT [6] for which learning rate, input sequence length, and batch size were optimized using random search.

As shown in Table 2, ULMFiT [13] and BERT achieve the highest accuracy on both experiments. However, it takes 68 hours to train ULMFiT on `Pat16` (around 20% of `Pat`), which is considerably slower compared to the other methods that take only a couple of hours (ranging from 3 to 9 hours)[6]. The reason is that building a language model from a very large corpus requires a lot of computing time. In addition, training Bi-LSTMs is generally slower than training CNNs.

The results of BERT are only provided for `Wipo-Alpha` as an initial comparison. It is not investigated further in this work because it was not yet available when the main part of this work was done. The language model is applied for classification using the commonly used approach[7].

Therefore, the next best model, Kim's CNN [18], is used for further experiments. 20 epochs can be trained within 3 hours on the `Pat16` subset. It is important to choose a model which can be trained relatively quickly, in order to further optimize it using hyperparameter optimization tools and apply other experiments and improvements.

4.3 Model Optimization

Word Embeddings First, the impact of different word embeddings on the accuracy is investigated. The three popular models, i.e. GloVe [22], Word2Vec [21] and FastText [2] are tested based on the different embedding sizes available

[4] `https://www.wipo.int/classifications/ipc/en/ITsupport/Categorization/dataset/index.html` (accessed May 4, 2019)

[5] `http://www.ifs.tuwien.ac.at/~clef-ip/download/2011/index.shtml` (accessed May 4, 2019)

[6] Specifications of the used machine: OS: CentOS Linux 7.5, RAM: 32GB Kingston HyperX Fury DDR4, CPU: Intel Core i7-7700, GPU: MSI GeForce GTX 1080 Ti Gaming X 11G

[7] `https://github.com/google-research/bert#sentence-and-sentence-pair-classification-tasks` (accessed June 24, 2019)

Table 2. Accuracy (%) of previously published methods for the reduced subset `Pat16` and a freely available dataset `Wipo-Alpha`.

Method	`Pat16`	`Wipo-Alpha`
Linear SVM [25]	68.8	41.0
NB [29]	65.5	33.0
FastText [16]	74.4	29.6
HAN [28]	79.7	49.3
Yoon Kim's CNN [18]	80.5	49.5
VDCNN [5]	76.1	41.3
ULMFiT [13]	**82.8**	49.7
BERT [6]	-	**53.4**

online. Furthermore, custom Word2Vec and FastText embeddings are trained on `Pat` containing 1.05 million patents.

Word2Vec and FastText utilize the skip-gram or CBOW algorithm to generate embeddings. Yet, FastText additionally considers n-gram sub-words as input instead of whole words as atomic units. Apart from the increased training time, it enables the model to embed unseen words. The word "Propene", for example, is unseen in the training corpus, yet FastText can assign a vector near the vector of the seen word "Propylene" which can be advantageous in domains with complex language like patents. Word2Vec lacks this ability and only offers the option to assign a random or zero-valued vector to unseen words.

There are many hyperparameters that can be optimized during the training of a custom word embedding [4]. In this work, only the default values are employed in order to ensure transferability. For testing the embeddings, a CNN was trained and evaluated on the `Pat16` subset for each embedding. Thereby, the embeddings were allowed to adapt during training time. The results of the evaluation are shown in Table 3. The accuracy on the first level of the patent hierarchy is used as evaluation metric.

The FastText embeddings trained on the in-domain corpus surpasses all other approaches. Yet, given sufficient embedding size (300) and training corpus the out-of-the-box embeddings reach satisfying results.

The same behaviour can be observed regarding the `Wipo-Alpha` dataset. Using custom FastText embeddings increases the accuracy from 49.45% to 52.26% with respect to the best pre-trained embeddings (GloVe, 300 dimensions, 1.9M words).

Hyperparameter Optimization In this section, the results of applying Hyperparameter Optimization by using BOHB [7] are investigated. In total, 30 runs of BOHB were executed on both the `Pat` and `Wipo-Alpha` dataset. The hyperparameters are optimized using only the training set of the respective dataset.

In the case of the `Pat` dataset, it is possible to completely randomize training and evaluation set in order to mitigate overfitting due to its size. Therefore before

Table 3. Results of different word embedding variations. The size of the vocabulary is given by the predefined embedding models.

Model	Dimensions	Corpus Size	Vocab Size	Acc. %
GloVe	500	840B	2.2M	80.05
GloVe	300	42B	1.9M	80.27
GloVe	300	6B	400K	79.77
GloVe	200	6B	400K	79.60
GloVe	100	6B	400K	78.85
GloVe	50	6B	400K	77.50
FastText (skip-gram)	300	600B	2.5M	80.50
Word2Vec (skip-gram, custom)	300	2.5B	350K	80.81
FastText (skip-gram, custom)	300	2.5B	350K	**81.35**

each iteration two disjoint, stratified subsets containing 20% of the complete dataset are sampled for training and validation.

As for `Wipo-Alpha`, a random 20% of the training set is used for validation in each evaluation.

Table 4. Hyperparameters ranges.

Parameter	Range
Learning Rate	$[1e^{-5}, 9e^{-3}]$
Batch Size	[16,256]
Dropout Rate	[0.1,0.6]
Number of Words	[10000,60000]
Regions Size	[1,15]
Number of Filters	[500,3000]
Sequence Length	[300,2500]

The examined hyperparameter ranges are illustrated in Table 4. Furthermore, Table 5 shows the best found parameters and their importance with respect to the prediction accuracy calculated by the fANOVA tool [15].

It can be seen that the importance of the parameters is similar in both datasets, except for Learning Rate, Dropout Rate, and Regions Size. It is hypothesized that the importance of these parameters differ between the two datasets mainly because of the differences in number of classes $(1,382/451)$ and their size $(250,000/75,000)$.

Table 6 shows that applying BOHB increases the accuracy by 1.59% on the 20% subset of `Pat` and by 2.76% on `Wipo-Alpha`.

Table 5. Optimal hyperparameters.

Parameter	Pat	Importance (%)	Wipo-Alpha	Importance (%)
Learning Rate	0.0002	21.9	0.0011	63.5
Batch size	64	1.3	128	0.1
Dropout rate	0.75	26.2	0.26	7.5
Number of Words	17567	6.7	23140	3.9
Regions Size	[4,5,6]	12.9	[6]	0.1
Number of Filters	2097	4.3	2426	7.9
Sequence Length	984	3.1	1431	5.5

Table 6. Hyperparameter optimization results.

Dataset	Before BOHB	After BOHB
Pat (20%)	60.84	62.43
Wipo-Alpha	52.26	55.02

Results The results after optimizing the CNN applied on the whole dataset are shown in Table 7. The optimized CNN (CNNopt) yields 6% higher accuracy than RNN-patent [24] achieving the current state-of-the-art results on Wipo-Alpha. Additionally, the results surpass a neural network based on BERT (cf. 2). The default CNN uses an upstream embedding (GloVe 300) and achieves an accuracy of 49.45%. The difference to the results before applying BOHB in Table 6 is caused by the usage of the custom FastText embeddings. Regarding Pat, optimizing the CNN increases accuracy from 62.01% to 65.43%, yielding a 6.71% improvement compared to the hSVM the baseline.

Table 7. Optimized results compared to base lines. Only available baseline results and published results for the datasets are given. Previously published methods of Table 2 are neglected due to runtime performance.o

Method	Pat	Wipo-Alpha
SVM	-	41.00
RNN-patent	-	49.00
hSVM	58.72	-
CNN	62.01	49.45
CNNopt	**65.43**	**55.02**

4.4 Discussion and Error Analysis

Figure 1 shows the accuracy on each level for both models, `hSVM` and `CNNopt`. `CNNopt` outperforms `hSVM` on all levels by a small delta although `CNNopt` is not a hierarchical method like the applied SVM.

The top-k results of `CNNopt` can be seen in Figure 2 providing an overview of the accuracy of each prediction. The accuracy reaches 95% when considering the top-6 predictions for all 1382 classes.

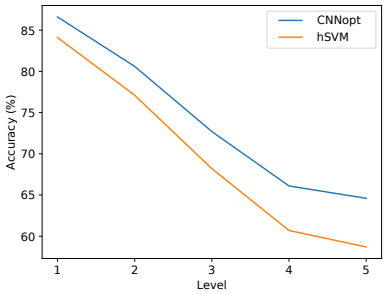

Fig. 1. Accuracy on each level for `hSVM` and `CNNopt` on `Pat`.

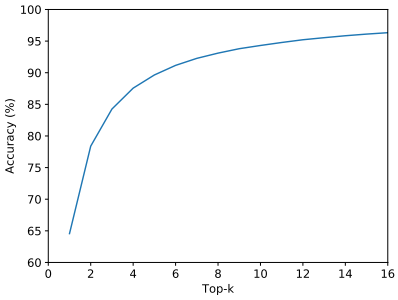

Fig. 2. Accuracy for top-k predictions of `CNNopt` on `Pat`.

Hierarchical Results As an experiment to present the hierarchical structure of `CNNopt`, the accuracy is calculated at each level regardless of the other levels. For example, if there are 100 examples in total, and in the first level 80 of them are classified correctly, then the first level accuracy is 80%. Afterwards, if 65 of those 80 are classified correctly in the second level, then the second level accuracy is 81.25% (65 out of 80 correct). This procedure is continued for all levels. Table 8 shows the results for `hSVM` and `CNNopt`, which indicate that `CNNopt` follows a hierarchical classification structure. Although the given labels are independent of one another and have no structure, its results are slightly better than `hSVM` at each level. Furthermore, a hierarchical CNN was implemented and evaluated but without considerable improvements, which is on par with the findings in this section.

Confusion Matrix The optimized network `CNNopt` achieves an accuracy of 80.5% on `Pat16` and 90.3% on `Pat` when considering only the first level. Initially, it was assumed that the data imbalance lowers the accuracy. However, the confusion matrix in Figure 3 highlights the main reason which is the similarity of classes. For example, the model shows the highest error rates between the two categories "Electricity" and "Physics" which share common characteristics.

Table 8. Level independent accuracy on `Pat` for investigating the dependence on the hierarchical structure.

Accuracy %	`hSVM`	`CNNopt`
First Level	84.1	86.6
Second Level	91.7	93.0
Third Level	88.4	90.2
Fourth Level	89.0	90.9
Fifth Level	96.8	97.7

The second highest error rate classes are the "Human Necessities" (health, hygiene, tobacco, food additives) and "Chemistry", which are similar as well. To ascertain this hypothesis, the top-2 accuracy was considered. As expected, the result improved from 80.5% to 95.5% on `Pat16`, while improving from 90.3% to 98% on `Pat`. Another hypothesis behind the reason the top-2 accuracy is much higher than the top-1 accuracy is that these patents are originally classified by multi-labels not only one label. In the single label dataset, only the most important class selected by some hidden heuristics is considered and the remaining are ignored. A patent may belong to different classes but with different levels of concentration (e.g. a patent may be discussing an electrical and chemical invention at the same time, but if it is concentrating more on the electrical part, then the leading class would be Electricity, while Chemistry would be a sub-class).

Error Analysis After inspecting and analyzing a sample of the common errors, three main types of errors can be identified:

1. Labels are classified correctly up to a certain level.
2. Labels are wrongly classified, but actually related.
3. Labels are wrongly classified.

Table 9 shows examples of the three types of errors and the description of the labels. The most frequent types of errors are the first and second ones. Using top-k predictions would mitigate these errors. However, this would lead to a multi-label classification problem, which would require modifications in the final layer of the network.

5 Conclusion

The task of automated classification of patents is far from being solved but with the growth of new patents filed from year to year it becomes more important every day. In this work we have studied the performance of several recent neural network models on the automated patent classification problem. It was shown that CNNs are a suitable choice in terms of accuracy and training/inference speed. We applied state-of-the-art hyperparameter optimization techniques to

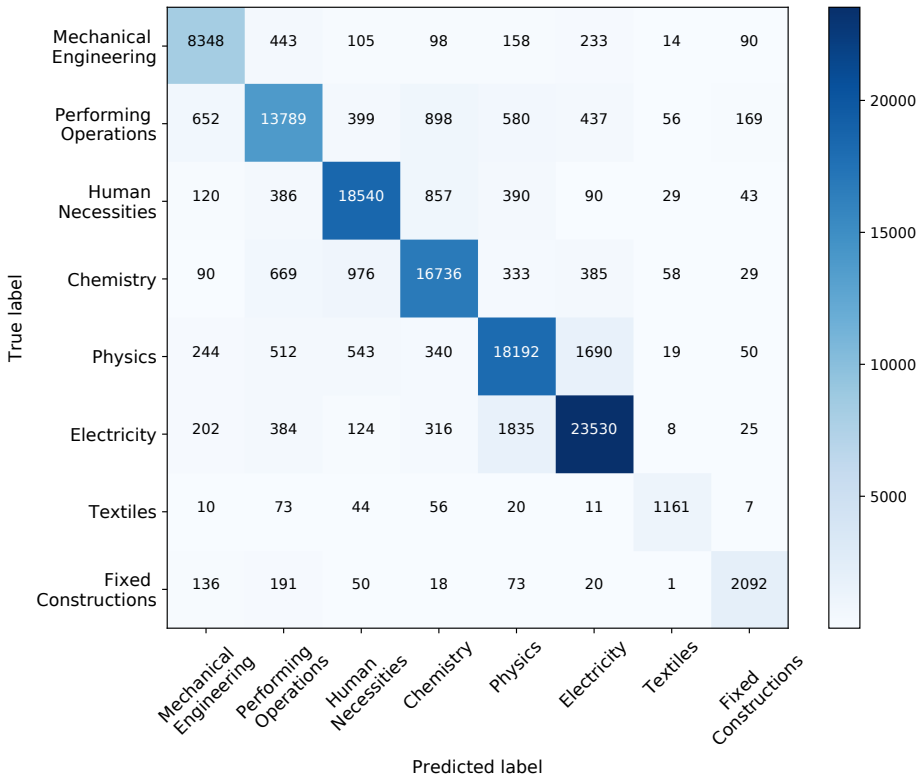

Fig. 3. Confusion matrix for the predictions of `CNNopt` restricted to the first level.

the problem and presented its effects on the accuracy. Furthermore, the results indicate that a complex hierarchical network may not be needed as the CNN learns the hierarchy of the labels by itself. The code for reproducing the experimental results is released as open source.[8]

References

1. Benzineb, K., Guyot, J.: Automated Patent Classification. In: Current Challenges in Patent Information Retrieval, pp. 239–261. Springer (2011)
2. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching Word Vectors with Subword Information. In: Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146 (2017)
3. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. NIPS 2014 Workshop on Deep Learning (2014)

[8] `https://github.com/lo2aayy/patent-classification`

Table 9. Different types of errors with labels description.

	Label	Description
predicted	A61K31/00	Medicinal preparations containing organic active ingredients
truth	A61K9/00	Medicinal preparations characterised by special physical form
predicted	G06F21/00	Security arrangements for protecting computers, components thereof, programs or data against unauthorized activity
truth	H04L29/00	Arrangements for network security (security arrangements for protecting computers or computer systems against unauthorized activity)
predicted	B65H1/00	Supports or magazines for piles from which articles are to be separated
truth	G07D1/00	Coin dispensers

4. Caselles-Dupré, Hugo and Lesaint, Florian and Royo-Letelier, Jimena: Word2Vec Applied to Recommendation: Hyperparameters Matter. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp. 352–356. ACM (2016)
5. Conneau, A., Schwenk, H., Barrault, L., Lecun, Y.: Very Deep Convolutional Networks for Text Classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, vol. 1, pp. 1107–1116. ACL (2017)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 4171–4186. ACL (2018)
7. Falkner, S., Klein, A., Hutter, F.: BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In: Proceedings of the 35th International Conference on Machine Learning, vol. 80, pp. 1437–1446. PMLR (2018)
8. Fall, C.J., Törcsvári, A., Benzineb, K., Karetka, G.: Automated categorization in the international patent classification. In: SIGIR Forum, vol. 1, pp. 10–25. ACM (2003)
9. Fall, C.J., Benzineb, K.: Literature survey: Issues to be considered in the automatic classification of patents. In: World Intellectual Property Organization, vol. 29, (2002)
10. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems?. The Journal of Machine Learning Research, vol. 15, pp. 3133–3181. (2014)
11. Gomez, J.C., Moens, M.F.: A Survey of Automated Hierarchical Classification of Patents. In: Professional Search in the Modern World, pp. 215–249. Springer (2014)
12. Grawe, M.F., Martins, C.A., Bonfante, A.G.: Automated Patent Classification Using Word Embedding. In: 16th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 408–411. IEEE (2017)
13. Howard, J., Ruder, S.: Universal Language Model Fine-tuning for Text Classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 328–339. ACL (2018)

14. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely Connected Convolutional Networks. In: 30th IEEE Conference on Computer Vision and Pattern Recognition, pp. 2261–2269. IEEE (2017)
15. Hutter, F., Hoos, H.H., Leyton-Brown, K.: An Efficient Approach for Assessing Hyperparameter Importance. In: ICML. JMLR Workshop and Conference Proceedings, vol. 32, pp. 754–762. PMLR (2014)
16. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of Tricks for Efficient Text Classification. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, vol. 2, pp. 427–431. ACL (2017)
17. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A Convolutional Neural Network for Modelling Sentences. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 655–665. ACL (2014)
18. Kim, Y.: Convolutional Neural Networks for Sentence Classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp. 1746–1751. ACL (2014)
19. Kowsari, K., Brown, D.E., Heidarysafa, M., Meimandi, K.J., Gerber, M.S., Barnes, L.E.: HDLTex: Hierarchical Deep Learning for Text Classification. In: 16th IEEE International Conference on Machine Learning and Applications, pp. 364–371. IEEE (2017)
20. Leopold, E., Kindermann, J.: Text Categorization with Support Vector Machines. How to Represent Texts in Input Space?. In: Machine Learning, vol. 46, pp. 423–444. Springer (2002)
21. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. In: CoPR. (2013)
22. Pennington, J., Socher, R., Manning, C.: Glove: Global Vectors for Word Representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543. ACL (2014)
23. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep Contextualized Word Representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 2227–2237. ACL (2018)
24. Risch, J., Krestel, R.: Domain-specific word embeddings for patent classification. In: Data Technologies and Applications, vol. 53, pp. 108–122. Emerald Publishing Limited (2019)
25. Steinwart, I., Christmann, A.: Support Vector Machines. 1st edn. Springer (2008)
26. Wu, S., Zhong, S., Liu, Y.: Deep residual learning for image steganalysis. In: Multimedia Tools and Applications, vol. 77, pp. 10437–10453. Springer (2017)
27. Xiao, Y., Cho, K.: Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers. preprint arXiv:1602.00367 (2016)
28. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical Attention Networks for Document Classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies pp. 1480–1489, ACL (2016)
29. Zhang, H., Li, D.: Naïve bayes text classifier. In: 2007 IEEE International Conference on Granular Computing (GRC 2007). pp. 708–708. IEEE (2007)
30. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, vol. 1, pp. 649–657. MIT Press (2015)