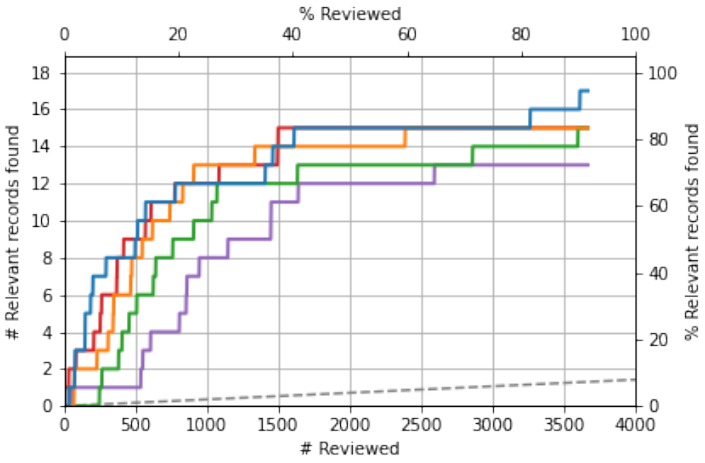
A comparison of performance between optimal neural networks and classical algorithms in active learning based text classification

J.J. Teijema

Supervisor: L. Hofstee MSc

June 2021

Keywords— Active learning, Systematic Review, Text Classification, Neural Network, CNN



Abstract

Deep learning is often used in text classification tasks for its efficiency and proficiency in modelling nonlinear processes. However, using this type of machine learning takes more time and processing power than using shallow learning algorithms. This study aims to determine if a combination of shallow and deep learning techniques can be used in increasing the classification performance for automated systematic review software. Since deep learning networks excel at finding difficult connections, this study aims to find situations in which they outperform shallow networks. To find these situations, simulations were run on a prepared dataset using different classifiers, switching from shallow to deep networks. While at first results showed no improvement, continued exploratory research provided results that support the hypothesis, as situations were found in which the neural network outperforms the shallow network classifier.

All data, code, and commands can be found at the following repositories:

- Study Repository - doi.org/10.5281/zenodo.5082962
- CNN Classifier Plugin - doi.org/10.5281/zenodo.5084887
- Model Switcher Plugin - doi.org/10.5281/zenodo.5084863
- Wider Doc2Vec Plugin - doi.org/10.5281/zenodo.5084877
- Data Repository - osf.io/8wa2n/
- Simulation Data Repository - osf.io/r45yz/

Contents

1 Introduction

A systematic review is a structured way of doing research by searching for and analysing large volumes of scientific literature. This type of review is useful for answering a central question using many sources and approaches, distilling them to find the answer(s) that combining these provides. This way, an unorganized myriad of research can be harnessed and utilized most effectively.

The work of systematically reading, grading and processing data is however very labor intensive and error prone (Wang, Nayfeh, Tetzlaff, O'Blenis, & Murad, 2020). This classification process can be greatly sped up, and even increased in quality, by utilising AI aided active learning systems. Such a system uses machine learning models to help users screen the records most likely to be relevant, and at the same time create a model more adept in selecting those relevant records.

These active learning systems can work well using shallow machine learning. These simple models are designed to be quick yet efficient and work well on the simpler tasks. Alternative to shallow machine learning, deep learning neural networks exist. These are often seen as the end all be all of classifying complex data, because of their strength in modelling nonlinear processes (Hopfield, 1982; Chollet, 2017). Using a neural network with the text classifier in the active learning process would then seem like a perfect match, and has been used many times so (Collobert & Weston, 2008; Hughes, Li, Kotoulas, & Suzumura, 2017).

To harness the advantages of both shallow and deep machine learning, there may be efficiency found in combining both methods to form a single AI classifier. This study aims to explore the feasibility, strengths and challenges found in combining shallow machine learning algorithms, deep leaning algorithms and the AI aided active learning environment.

To increase efficiency in explaining problems and approaches in this study, appendix C lists commonly used terms within ASReview and provides explanations.

1.1 ASReview

This study will be working with ASReview (van de Schoot et al., 2021) for its systematic review software. ASReview classifies data using classical machine learning algorithms like Naive Bayes, SVM, Random forest and more. These methods work fast and efficiently, achieving high recall rates (see C.1) in little time with low computing power costs.

The ASReview pipeline[1] contains the following steps:

1. Feature extraction: This step turns the input data into a more abstract representation usable by the machine learning algorithms.
2. Training: A classifier model is trained on the labeled data and estimates a relevance score for all unlabeled data. This training only starts when the training of a previous model has finished and there is new labeled data available as a result from manual labeling by the user. This means that after starting the ASReview labeling process, there is almost always a new model training in the background, until this process is finished.

[1] Active learning for Systematic Reviews at https://asreview.readthedocs.io/

3. Improving: The user labels records from the list of yet unlabeled records, sorted by the classifier on likeliness to be relevant. When a user labels a record, it is then added to the store of known labeled data both relevant and irrelevant. The user continuously labels the most relevant records, while a new model is training.
4. Repetition: The improvement and training steps are repeated until the model is accurate enough to label all the data correctly according to the user, or the user believes that all relevant records have been found.

This system, called an AI aided active learning system, uses AI and users in conjunction to form an efficient method for systematic reviews.

1.2 Why Neural Networks

The most prominent drawback of shallow networks is apperent when used to find complex and hidden connections. Deep learning networks (i.e. neural networks) are better at finding complex connections within data when compared to shallow networks such as the classical machine learning algorithms currently applied in ASReview. (Rolnick & Tegmark, 2017), for instance, proved that it is exponentially easier to approximate sparse multivariate polynomials with neural networks when compared to shallow networks performing the same task. The term deep learning comes from the depth in layers neural networks have. Where shallow networks only have one, or at most two layers, a deep learning network can have many layers, only restricted by computing power available. These deep layers are where the complex connections are found.

For a long time, shallow networks were the golden standard in classification. First Naive Bayes, a probabilistic classifier, then more advanced machine learning techniques like SVMs, decision trees, and gradient boosting machines. These methods however proved to be hard to scale and did not provide good results for the more complex problems.(Chollet, 2017) The complex connections between data points found by neural networks can be essential for certain classification tasks such as text processing, image recognition, and more.

One of those complex problems can be found in the core functionality of ASReview. The text classification problem is older than neural networks, but in recent years dominated by them. The website Kaggle hosts machine learning competitions often, and where shallow networks used to win prizes only years ago, today neural networks have the upper hand. Seeing this, it would make sense to consider neural networks for use in ASReview.

In the case of the hypothesis it makes even more sense to use deep learning methods. The shallow networks perform extremely well on the records that are connected in a straightforward, shallow, way and it would make no sense to change them. The records connected in a more complex manner are likely to need deeper networks, calling for the use of neural networks. This is especially the case when it comes to plateaus (see 1.3).

1.2.1 Neural Network challenges

Complexity

While neural networks might be the preferred model for nonlinear classifying, they do bring with them some drawbacks (Montavon, Orr, & Müller, 2012). When compared to classical algorithms they need more input data, take more computing power, and more simulation time to complete training, as training these networks is a more complex process. The original neural network in ASReview trains in 35 epochs, each taking multiple seconds. This means that each time this network is trained, it can take multiple minutes, increasing with each newly added labeled record.

Feature engineering

A big strength of neural networks is that they do not require feature engineering. This means that where shallow networks need the input data to be carefully processed beforehand, a neural network does not need so. The requirement of precise feature engineering made the shallow networks fragile and hard to work with. This manual work for the data engineer, finding good representations of data for the shallow network to work with, was greatly reduced by the rise of deep learning as all features are learned automatically during training.

ASReview, however, already has strong feature engineering. The feature extraction methods like doc2vec and tf-idf and the balancers implemented have been proven to work very well with the shallow machine learning on data classification problems. This means that this special property of neural networks is redundant, and shallow networks are ahead in this aspect as neural networks still spend a considerable amount of computing power on this step, which is thus wasted for ASReview.

Empirical versus theoretical nature

Finally, where the shallow networks have a strong mathematical and theoretical basis, the deep learning algorithms are hands-on and are proved more empirically than theoretically. Herein lays a strength and a weakness. On the one hand, the network works well on complex tasks without being "restrained" by under laying maths and fragile approaches, making them easy to implement and use.

On the other hand, it makes them hard to get right. Neural networks seem to require a certain intuition, hard to capture with rules. "How many layers and nodes do I need" is a question answered almost exclusively with either "get a feel for the data" or "try them all" (Stathakis, 2009). This results in an optimal neural network being a loose term and hard to create.

1.2.2 Hyperparameters

Yet another challenge of the research question lays in the "optimized" part of the neural network. A neural network can work extremely well, or not at all, entirely depending on if the hyperparameters of the network are well thought out. The neural network used for proving that outperforming the classical network is possible should have optimal parameters for the setting, without being over-fit for the specific application.

A neural network contains two sets of parameters. The first set, the internal parameters made from weights and biases, are found and set during the training phase of the network. They are directly related to the function of the neural network and

are applied to the input layer in such a way that the combination of all these inputs, weights, and biases result in a desired output.

The second set, the hyperparameters, are the settings dictating the form and behaviour of the network. The hyperparameters set the type of network from simple dense networks to advanced networks using backpropagation, temporal sequences, and memory neurons.

Some of these second set hyperparameters do not get in contact with the input and output of the network aren't present after training. They do however influence how efficiently the network is trained and how well it performs. Optimizing this parameter set is a crucial step in creating an usable well fitted network.

1.3 Plateauing

The data that ASReview works with are abstracts from literature found after searching databases for information. The distribution of records can form clusters if the dataset spans multiple topics. If the classifier has found many records from a single cluster, it can be over-fit to find records from a different clusters as they are not like the found records. Only when a record from the new cluster is found can the classifier start finding other records from the same cluster. These clusters can create some difficult situations during classifying.

A common trend in ASReview simulations is that the recall curve seems to take a logarithmic form. The first half of the records can almost always be found easily, whereas the second half and especially the last few percent of records take significant effort. Figure 1 from the ASReview benchmarks shows the recall curve for multiple different classifiers over a benchmark dataset, and the behaviour is clearly visible.

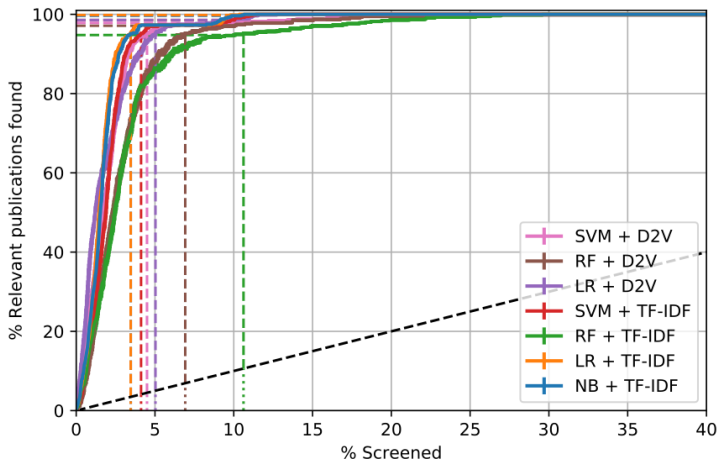


Figure 1: Recall curve example for different classifiers and feature extractors.

This logarithmic trend is caused by harder to find records. The system first picks the most closely related records, resulting in a quick rise in found records. The more distantly related records are then harder to find when the algorithm is over-fit.

When a not yet found cluster is removed relatively far away from the current identified cluster, a plateau forms. This plateau represents a stagnation in found records until new information is found. This new information can be found per chance or because of unexplored weak connections.

Neural networks are by design more capable of finding hidden connections, and might find hidden clusters faster than the classical algorithms if the clusters have deeply hidden connections. That is, in the likely case of these plateaus being caused by the yet undiscovered records being more distantly related to the thus far labeled data.

1.4 Hypothesis

Since the relevant records are chosen based on the latest trained model, it is important to have the new model trained and available *before* the user is done labeling the record. Especially in the first records, the information a single labeled record can provide is instrumental in efficiently providing the user with the best next record to label. In the later stages of the labeling, the importance of training a new model with every new labeled record decreases. This is the difference in added information between e.g. 3 and 4 labels or between 45 and 46.

The average labeling time for users is 40 seconds, as determined by ASReview user data. This means that for the system to be efficient at the start of the process, either the model needs to train in under 40 seconds, or the extra simulation time increases performance more than having more data. As such a performance increase is unfeasible in the first stages, it would be unwise to use a network that trains for more than 40 seconds from the start. However, at some point the extra training time might increase the ASReview performance more than having slightly more labeled records. However, note that the gain lines in the figure 2 are speculation.

This study will attempt to prove that there are situations in which an optimized neural network will outperform the algorithm at some optimal point on the recall curve (see C.2). The hypothesis follows:

- There exists a situation in which an optimized neural network can outperform the classical algorithm, if applied at the optimal point in the recall curve.

The precise location of this optimal point will be an estimation in this study. The estimation is derived from those precondition criteria that increase the strength of a neural network, such as having increased simulation time and complexity, making better use of having more information (more labeled data) than classical algorithms and being better in solving abstract classification problems.

By finding the optimal switch point on the recall curve and switching out the classical algorithm with a neural network, the hypothesis can be tested. If the performance significantly increases over the original run without switching models, then the hypothesis can accepted.

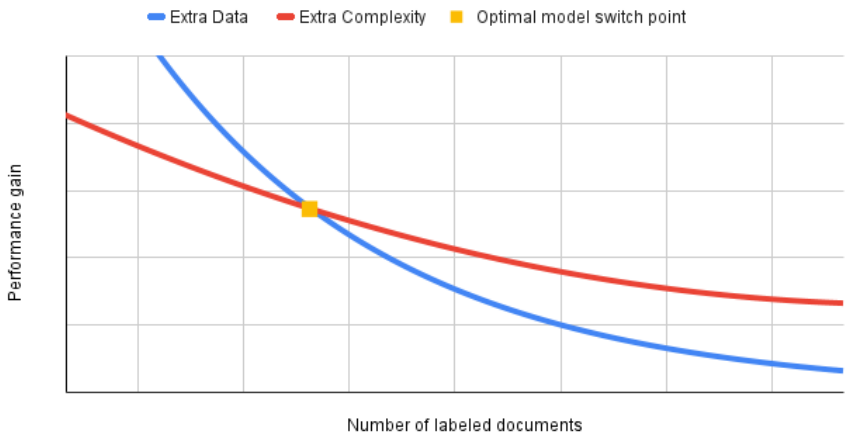


Figure 2: Theoretical performance gain from extra data and extra training duration.

1.4.1 Sub hypothesis

The hypothesis this study proposes, depends on the notion that the shallow networks have more difficulty with clustering than deeper networks. With this in mind together with knowledge of clustering, a sub-question of the hypothesis forms;

- Shallow machine learning classifiers such as the Naive Bayes face a bigger performance impact than deep learning networks as a result of clustering in datasets.

During simulations, new records are found based on identified records. When the simulation is cut off somewhere during the process, the thus far found records are likely in a cluster, as those records are found first. Less information will be known about records in undiscovered clusters.

ASReview offers the option to specify a number of labeled records to be set for a simulation, in an attempt to mimic prior knowledge for the simulation. When specific records are not provided for the simulation, these records are selected at random. They are therefore likely to be located across several clusters. This changes the characteristic of the simulation in that, under normal circumstances, a shallow network would build its knowledge from records that are in the same cluster.

In fact, the performance of the shallow network, normally hampered by its inability to find different clusters, is artificially helped by building prior knowledge from randomly selected records across several clusters, while this is exactly what neural networks attempt to use to outperform them.

2 Methodology

2.1 The Brouwer Dataset

To evaluate the performance of ASReview and the new classifiers, a dataset is needed. For this study, the data from a systematic review was used (Brouwer et al., 2019). The systematic review in question reviewed data regarding different theories for depression relapses. Since the review analysed 5 different theories, the data found is a combination of records from those categories.

This systematic review processed 50936 records and identified 63 relevant ones. The ratio of relevant to irrelevant is 808.5 to 1. The dataset contains information in different theory subsets; Behaviour, Cognitive, Diathesis, Personality, Psychodynamic.

2.2 Methodology overview

Implementing a neural network, as flexible and manoeuvrable as it may be, needed a structured approach. Figure 3 outlines the road map for the realization of this study. Each step will have a more in-depth explanation in this methodology.

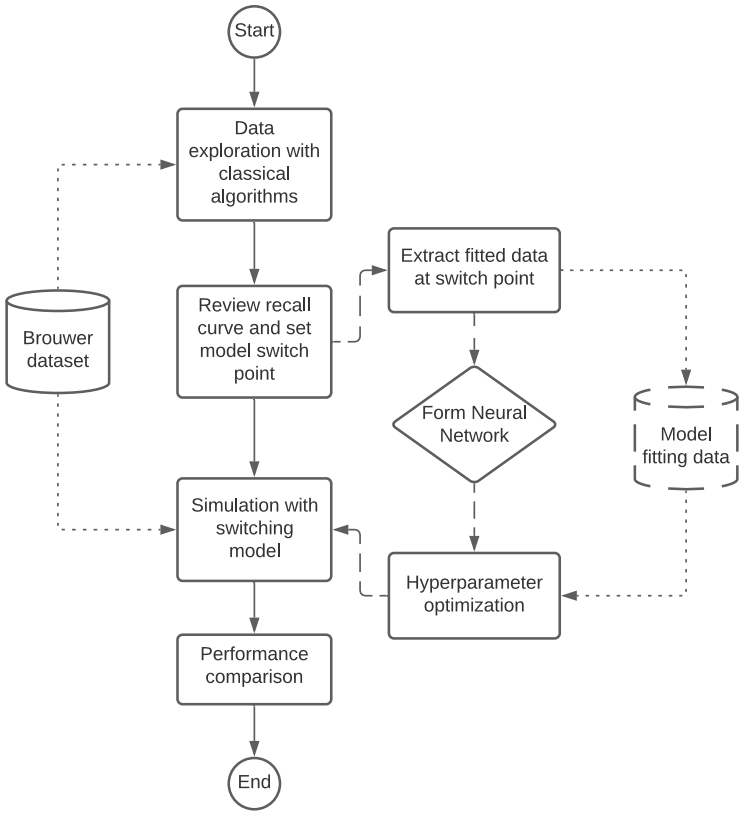


Figure 3: Research approach road map.

2.3 Data exploration

During data exploration, a simulation was run using the dataset in order to measure performance and visualize the recall curve. The dataset was analyzed using the simulation mode found in ASReview. The dataset was simulated first with Naive Bayes, the currently best performing classifier implemented in ASReview based on ASReview benchmark runs. The simulation ran using TF-IDF as feature extraction, and the standard setting for query and balancing strategies. The defaults for this version were MAX querying and Double Balancing.

All simulations are run on the Google Colab Pro platform, using ASReview Release v0.17 (2021-04-28). More information on how the simulations were run and with what exact settings can be found in the GitHub Repository accompanying this study, and in appendix B.

2.4 Model switch point

The model switch point is a condition to meet for ASReview to be instructed to switch from the classical machine learning model to the optimized neural network, and should be at a moment where the situation is optimal for the performance of the neural network. The reasons a neural network might outperform a classical algorithm have been explained throughout the theoretical framework.

Based on theory and found phenomenon in the data exploration, the optimal location for switching between a shallow and a deep learning network should be where model complexity is preferred over a model finding simple connections. For this study, the optimal location for the neural network is found at the beginning of a plateau, while having enough labeled data to fully utilize the deeper nature of the network.

2.5 Fitting data

To optimize the neural network with real data, fitting data was extracted from the classifier at the model switch point. This fitting data was then used to optimize the neural network. By using data from this point, the neural network can be expected to work optimally for its application.

2.6 Neural network creation

There are many types of neural network, and which type to use depends on the input data type, the outcome variable, and the task.

This study proposed a convolutional neural network for its text classification as these types of neural networks are often successfully used in text classification tasks (Collobert & Weston, 2008; Hughes et al., 2017). Convolutional neural networks originated as image processors called neocognitrons, but this type of neural networks can be applied very successfully on NLP tasks, while having a small computational footprint.

Convolutional neural networks, or CNNs, consists out of convolutional layers with a shifting kernel. In this study, the kernels and layers are one dimensional where image

processors are often two or three dimensional. This one dimensional shape matches better with the records used as input data. The text for each record can be used as a time series, with words appearing one after the other from the beginning to the end of the document. A lot of information is retained this way as not only words themselves can carry meaning, but word order can do so too.

Convolutional layers are specialized and efficient neural networks, much more so than standard dense layers. In dense layers (often called fully connected layers), each neuron is connected to every neuron in the layer before, with its own parameters. This makes them expensive to compute. Convolutional layers are only connected to a few neighbouring neurons, and the weights are the same for each connection. This makes them cheaper to compute than dense layers.

These local connections are useful for extracting information from input data where features are locally related, such as images or words. This makes convolutional layers strong in image and text related neural networks, and thus for the ASReview application.

The proposed CNN was based on the (Hughes et al., 2017) text classification study with similar goals as ASReview. In that study a convolutional neural network was created for medical document classification, and achieved accurate results. The neural network implemented in this study has the same layers and structure, but different layer sizes and uses doc2vec instead of word2vec, as the input is an entire abstract and not just loose sentences.

The resulting convolutional neural network is made up out of a combination of separable convolutional layers (SeparableConv1D), ReLu activation layers, Dropout layers and final Dense (fully connected) layers (Teijema, 2021). Following direct advise in (Collobert & Weston, 2008), separable convolutional layers are used, promising a lighter, faster, and better performing model. These layers separate certain steps, significantly reducing trainable parameters. The precise architecture can be found in figure 4.

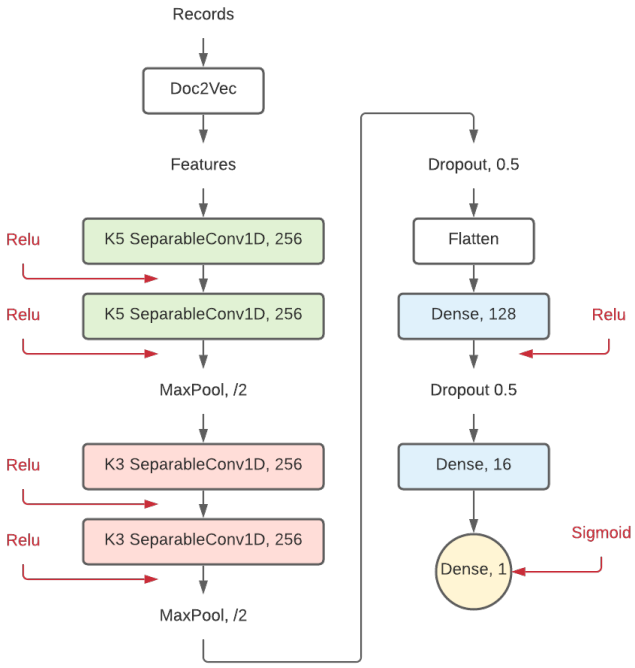


Figure 4: Proposed CNN

- SeparableConv1D - These layers will be used to detect patterns and connections within the records. The ReLu activation accompanying these layers has been proven to be beneficial for training deep neural networks. (Glorot, Bordes, & Bengio, 2011)
- Dropout - The Dropout layers are used as prevention for over-fitting by setting half of the nodes to 0 during each training step. Without dropout, a node can correct behaviour for another node during training. This corrective behaviour can lead to over-fitting because these fused nodes do not generalize to unseen data. Dropout prevents this from happening and thus reduces over fitting (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).
- MaxPooling1D - These layers are used to both reduce the network data size and generalize patterns found by having kernels in following layers look at relatively more data while staying the same size.
- Dense - At the end of the CNN, two Dense layers are set up, finishing the network. These layers connect all pattern together, which doesn't happen in the local-only convolutional layers.

2.7 Hyperparameter optimization

Optimizing the new neural network was done using a grid optimizer. Multiple settings were selected for trail runs, and each combination is tested 3 times. While this does

take a considerable amount of time, it results in approaching the optimal selection of hyperparameters fit for this neural network, for this situation.

The model is set to 100 epochs, but will not train that many. Early stopping based on model loss values has been implementing, stopping training when no more significant loss decrease (0.025 to be precise) has been found in the last 10 epochs. The early stopping system then restores weights for the earliest epoch with this performance. This decreases over-fitting.

The batch-size for the model was set to adjust based in the size of the input data. This way, the model adjusts better to each step in the recall curve.

There was no modified class weight used as ASReview already has an integrated balancing functionality.

2.8 Model switching

Model switching can be done manually. If the new models are run from the model switch point with all thus far labeled data as prior knowledge, then this would be the same as switching during classification. This is because models are trained from scratch each iteration, and only the list of labeled records is passed along.

To achieve the model switching simulation results, record row IDs were extracted from labeled data, and used as prior data for the simulation.

To perform this kind of model switching automatically, a model switching plugin for ASReview was created, the ASReview Model Switcher. Hosted on GitHub, the plugin is installable using a single command. The new switching model is available as a new classifier model, using -m SVM-NN, and switches models at a set switch point. The plugin is modelled after the new model template provided by ASReview.

The software has a model switching base class, and is then easily implemented with a child class, using the super class functions. The switching point is manually set with an argument in the __init__ function. This means that adding new switching models can be done easily, with only a few lines of code.

On top of the model switching software, the new convolutional neural network is also available. Used with -m power_cnn in ASReview after installing the plugin.

2.9 Performance metrics

Although the hyperparameters of the developed convolutional neural network were optimized on classification accuracy, the comparison performance between classifiers was measured using the recall curve, and the amount of relevant records found after 4000 processed records. This number was selected for being close to the WSS%100 for the Naive Bayes classifier. If the new classifier being compared was WSS@%100 under 3909 records then performance was better than the Naive Bayes classifier, and if WSS@%100 was found after 3909 or not found at all, then performance would be worse. There was no reason to test performance after 4000 records as this would not be relevant to the hypothesis and research questions.

2.10 Simulations and Outcome

All simulations were run using Google's Colab Pro platform, using a GPU infrastructure specifically designed for running machine learning loads. The runs were executed in parallel on multiple Nvidia Tesla K80 GPUs. Appendix B has code and information on how the simulations were run in parallel using this platform.

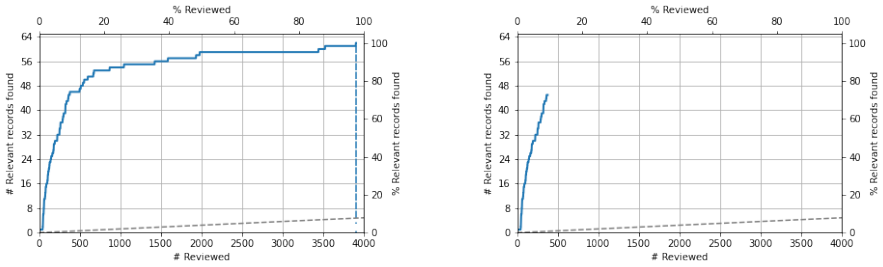
The simulations were run over the entire dataset using the standard Naive Bayes classifier, for data exploration purposes. Then simulations were run from the model switch point using the random priors for the second hypothesis. As explained in 2.8, the expected result here was the shallow networks outperforming the more complex deep networks. Finally, simulations were run from the model switch point onwards using priors as determined by initial Naive Bayes simulation.

3 Results

3.1 Simulations

First, an exploratory simulation was run. This run is used as a baseline for comparison with other classifiers. The simulation was run with the currently best performing classifier, Naive Bayes. The results of this first run are shown in figure 5a, and the fitting data was extracted at the end of the recall curve in figure 5b.

Out of the 50936 total records, Naive Bayes finds all records after 3909 screened records. This means that this classifier saves 92.4% of work over manually screening all records. At 379 records screened, having found 46 relevant records, a clear drop in classification performance can be seen.



(a) Full Recall curve

(b) Data extraction point

Figure 5: Fitting Data Extraction

Next, different models were ran from record 379 to record 4000, using 46 relevant and 333 irrelevant documents, but using random documents. This simulates a model switch point at 379, in accordance with the theoretical switch point from the method. The results are shown in figure 6.

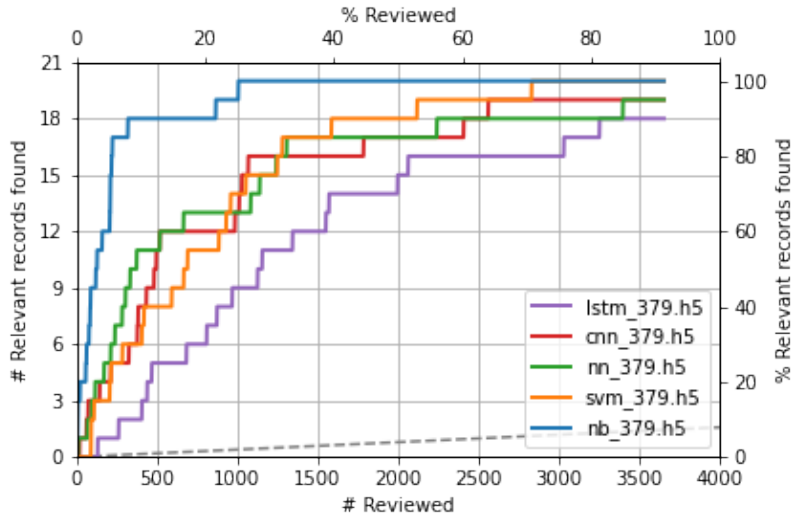


Figure 6: Simulations using random prior, as measured from the model switch point

Finally, all models ran from the model switch point, but this time using the exact simulation state. Again, the simulations range from 379 to 4000. Figure 7 shows the performance of all models.

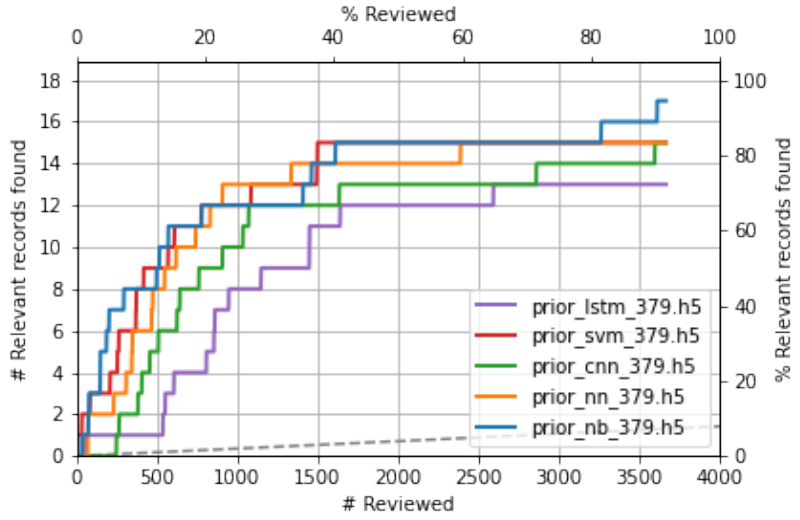


Figure 7: Simulation using model switching, as measured from the model switch point

4 Discussion

4.1 Main findings

This study aimed to explore feasibility found in combining shallow machine learning algorithms, with deep leaning algorithms. The results show no evidence for increased performance over using only the original shallow classifier. This is not in line with expectations and means that the hypothesis can not yet be accepted. That there is no evidence suggesting that switching models alone is beneficial, with the currently implemented classifiers and feature extractors.

A reason behind not having found increased performance could be that while the other classifiers were appropriately applied, none are powerful enough to increase performance over the shallow classifier. Another option is that the new convolutional neural network was not optimized enough for it to outperform the Naive Bayes as a result of being structured wrongly. Or, it could be that while the convolutional neural network is performing well, the feature extraction is not.

The simulation with random priors does show that the Naive Bayes classifier has more benefit from random priors than other classifiers in this situation, indicating that Naive Bayes might have more cluster related performance issues than other classifiers.

4.2 Limitations

For this study only 3 types of neural networks were tested, but for this project to give more insight into the performance of different networks, more types of neural networks should be explored and evaluated.

Deep learning with neural networks is an empirical science, and concluding a neural network to be "optimized" based on theory is, as a consequence, not feasible. Empirical science requires many tests, cross validated many times over. The scale of this study did not allow for a true answer to the question of using neural networks in ASReview. Only an approximation or direction towards the answer was given.To answer the question with certainty, more research and more simulations are needed.

4.3 Potential research areas

If the Naive Bayes classifier indeed has a bigger negative performance impact as a result from clustering than other models, there may be much untapped potential there. If a solution exists that reduces the gap between clusters for example, then performance might rise far above thus far observed.

For this study, the model switch point was chosen based on certain factors, but not extensively tested. A follow up study, looking into the effect of switching models with different switching conditions, should give more insight into the workings and effects.

Finally, when comparing Naive Bayes and the CNN recall curves, the length of the plateaus stands out. Naive Bayes is very quick in detecting records easy to detect, but encounters the longest plateaus out of all classifiers. A possible alternative to the model switch point is switching from shallow network to deep learning network after a plateau has been detected, and then switching back after a new record has been found.

Of course, plateau detection has not been implemented yet, but this might even more efficiently utilise the strengths of deep learning networks and shallow networks. Had this study not been restricted on time, this idea would have been explored fully as it has a significant potential performance increase.

4.4 Exploratory research

Results of this study were mostly not in line with expectations according to the hypothesis. One conclusion to draw from this, is the importance of hyperparameters for neural networks. It was very likely that the reason for the lack of performance increase came down to this. To this end, further exploratory research was performed exploring more branches of hyperparameters looking for adjustments that do increase performance. Following trial and error with the proposed and implemented convolutional neural network, there have been situations found in which the hypothesis can be accepted. With these settings, the convolutional neural network beats the Naive Bayes classifier by over a thousand screened records, even without being fully explored in potential.

The change that increased performance most, was adjusting the feature extraction hyperparameters. This was not explored during the study trails. This change resulted in the convolutional neural network outperforming the Naive Bayes classifier. The main difference in performance was gained in increasing the doc2vec vector size. The results of this run can be seen in figure 8.

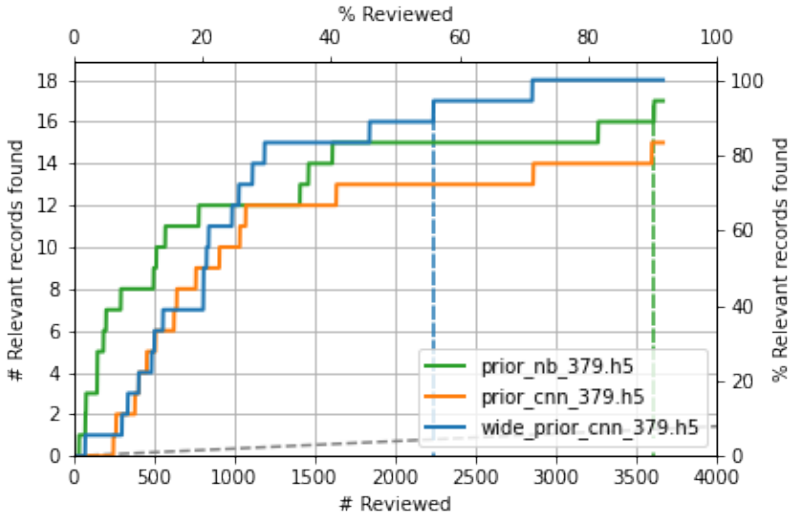


Figure 8: Simulation using new feature extraction settings

Recreating this results can be done using the wide_doc2vec plugin. This is a simple copy of the original doc2vec feature extractor, with a wider input vector size.

These results indicate that there is potential in switching models halfway, and show that there exist cases in which deep learning based classifiers are the better choice of model. While these results are not conclusive, they do call for continued research into implementing new neural networks, and feature extraction methods, for ASReview. Especially into the way the feature extractions are used for neural networks.

Worth noticing is the initial stagnation of performance directly at the start of the simulations for the convolutional neural networks. So far, no reason for this initial lag in performance has been found, but this might be a result of software error, and not performance.

4.5 Conclusion

The aim of this study was to find a combination of shallow and deep learning neural networks that could outperform the Naive Bayes classifier. This combination was, in the final exploratory research, found. The combination of Naive Bayes and a new proposed convolutional neural network outperformed Naive Bayes.

Based on that result, this study calls for the continued exploration of neural networks in AI aided active learning based text classification.

References

Brouwer, M. E., Williams, A. D., Kennis, M., Fu, Z., Klein, N. S., Cuijpers, P., & Bockting, C. L. (2019). Psychological theories of depressive relapse and recurrence: A systematic review and meta-analysis of prospective studies. *Clinical psychology review*, *74*, 101773.

Chollet, F. (2017). *Deep learning with python*. Simon and Schuster.

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on machine learning* (pp. 160–167).

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 315–323).

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, *79*(8), 2554–2558.

Hughes, M., Li, I., Kotoulas, S., & Suzumura, T. (2017). Medical text classification using convolutional neural networks. *Stud Health Technol Inform*, *235*, 246–50.

Montavon, G., Orr, G., & Müller, K.-R. (2012). *Neural networks: tricks of the trade* (Vol. 7700). springer.

Rolnick, D., & Tegmark, M. (2017). The power of deeper networks for expressing natural functions. *arXiv preprint arXiv:1705.05502*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958.

Stathakis, D. (2009). How many hidden layers and nodes? *International Journal of Remote Sensing*, *30*(8), 2133–2147.

Teijema, J. (2021, Jul). Asreview cnn 17 layer model plugin. doi: 10.5281/zenodo.5084887

van de Schoot, R., de Bruin, J., Schram, R., Zahedi, P., de Boer, J., Weijdema, F., ... others (2021). An open source machine learning framework for efficient and transparent systematic reviews. *Nature Machine Intelligence*, *3*(2), 125–133.

Wang, Z., Nayfeh, T., Tetzlaff, J., O'Blenis, P., & Murad, M. H. (2020). Error rates of human reviewers during abstract screening in systematic reviews. *PloS one*, *15*(1), e0227742.

A Personal remarks

A.1 On the Brouwer dataset

Since the Brouwer dataset can be assumed to behave similarly to the planned research in the future on the same topic, this study suggests that switching models could be beneficial for finding relevant records quicker than using the currently implemented used methods.

A.2 On Neural networks

Neural networks almost feel like cheating. Nature, in its infinite complexity, seems strongly unwilling to let itself be captured by only math (although I am not saying it won't be possible). By allowing neural networks to be equally unrestrained and chaotic it sets them free to capture more closely the whims of nature. The funny thing is, getting a neural network to work right takes certain human intuition not captured by rules or governed by rules not recognized yet. There are some interesting parallels to be drawn there.

I am not fooled by the notion that neural networks mimic our own brains. Still, neural networks being able to approach tasks with an uncanny human-like intuition, because they have a natural touch, makes you wonder about the structures behind our reality. For this, I personally believe the future of deep learning to grow even more towards our own nature. Neural networks not designed but formed by evolutionary deep learning algorithms. An evolutionary artificial neural network optimized by its genetic code. It will take some more computing power than we currently have though, which for some reason, always seems to be the case in the machine learning field.

Finally, I believe that the perfect solution for ASReview is an Evolutionary Artificial Neural Network, that grows alongside the active learning, with each new labeled record. Sadly, research into this field is not there yet, and neither is the processing power. We'll have to stick with gradient decent for now, and some human intuition.

A.3 Useful resources

While scientific literature formed the basis of this study, certain other resources proved to be incredibly useful. The first being the GitHub repo github.com/buomsoo-kim/Easy-deep-learning-with-Keras. This Repository contains many examples of different neural networks, implemented and working well. The other is a collection of resources about Deep Learning collected and maintained by the LSESU Data Science Society satlse.github.io/resources/deep-learning/.

B Running multiple simulations in parallel on Google Colab

The simulations for this study were run in a Jupyter Notebook using Google Colab Pro. The most important time saving feature used during this study was running simulations in parallel. While the simulation and training speed could not be increased further than the already impressive speeds reached on Colab, a quirk on this platform allows for running an unrestricted amount of processes as background threads, without decreasing individual speeds. Colab runs all threads on a separate processing core, and thus threads do not slow each other down. As a bonus, the code seemed to become more resistant to user disconnects.

The following code allows for multiple processes:

```
from threading import *
import os
os.system('ls -l')


class App1(Thread):
    def run(self):
      os.system("console command to be run")


class App2(Thread):
    def run(self):
      os.system("parallel command to be run")

app1 = App1()
app2 = App2()

app1.start()
app2.start()
```

As the threads are run in the background, there is no way to see console output. Checking if threads are still alive can be done with the following code, but note that this cell runs continually and thus blocks other cells from running as long as the threads are alive. Furthermore, stopping this cell while the threads are still alive has a slightly chance of halting all running background threads, and corrupting all state files. Use with care!

```
import time
from IPython.display import clear_output
while (app1.is_alive() or app2.is_alive()):
  clear_output()
  print("\n", "Thread 1 is alive: ", app1.is_alive(), "\n",
        "Thread 2 is alive: ", app2.is_alive(), "\n",)
  time.sleep(8)
```

C Terms and Metrics

To increase efficiency in explaining problems and approaches in this study, the following terms are relevant.

C.1 Recall

Recall is a metric often used in other metrics. It is the percentage amount of relevant records found, either by manual work or by the ASReview software. If there are a 100 relevant records, and 95 are found, then the recall at that point is 95%.

C.2 Recall Curve

The recall curve is a representation of the found relevant records (the recall), in comparison to either the absolute or relative amount of screened records. See figure 9 for an example with multiple recall curves for different classifiers.

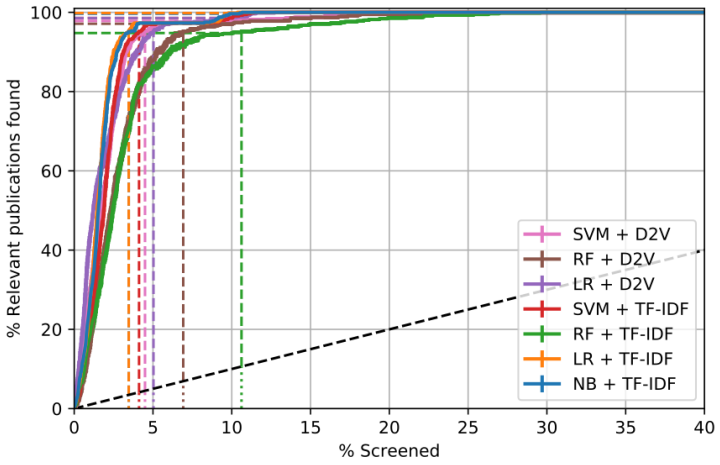


Figure 9: Recall curve example figure, found on https://asreview.readthedocs.io/en/latest/guides/simulation_study_results.html.

C.3 Work Saved over Sampling

WSS or Work Saved over Sampling, is the percentage reduction in amount of records that need to be screened using the active learning system over screening records randomly. When talking about a value for WSS@95%, it represents the WSS value at the moment where the recall is 95%, and thus when 95% of the relevant records are found.

C.4 Relevant Records Found

RRF or Relevant Records Found. It measures the total amount of records screened after having found a set amount of relevant records. It is the opposite of WSS which measures relevant records of total screened. RRF@30% represents the total screened records after having found 30% of the relevant records.