

LIVERPOOL HOPE
UNIVERSITY

Evaluation of
Kolmogorov-Arnold Networks
in
Natural Language Processing:
A Case Study in Sentiment Analysis

Kaloyan Tsankov

A thesis submitted in partial fulfilment of the conditions of the award of the degree
MSci Artificial Intelligence

School of Mathematics, Computer Science and Engineering
Liverpool Hope University
Supervisor: Dr Brett Drury

August 2024

Declaration of Originality


This thesis is submitted in partial fulfilment of the requirements for the degree of MSci offered by School of Mathematics, Computing and Engineering, Liverpool Hope University.

I, Kaloyan Tsankov, hereby declare that this dissertation is my own work, and builds upon my previous work titled "Sentiment Analysis with KANs," [\[17\]](#) which was submitted as part of the COMM013AZ2023/4 - Artificial Intelligence (2023/4) module. The coursework provided a foundational basis for the research and development presented in this dissertation. All other sources of information have been duly acknowledged.

I confirm that I have obtained informed consent from all people who participated in this work and received ethics approval as appropriate to my dissertation.

I confirm that the word count for this dissertation including title page, declaration, abstract, acknowledgements, table of contents, list of illustrations is 7123 words.

I also permit Liverpool Hope University to store a copy this dissertation in a public archive.

Signature: 

Date: 30th August 2024

Contents

1	Introduction	3
2	Motivation	3
3	Literature Review	4
3.1	Trends in Sentiment Analysis	4
3.2	Data Augmentation	4
3.3	Convolutional Neural Networks for Sentiment Analysis	5
3.4	Kolmogorov-Arnold Networks (KANs)	6
4	Methodology	7
4.1	The Data	7
4.2	Easy Data Augmentation	7
4.3	The Neural Network	8
5	Implementation	8
5.1	Hardware	8
5.2	Technical Approach	9
6	Challenges	10
7	Experimental Results and Evaluation	11
7.1	Task evaluation	11
7.2	Impact of EDA	11
7.3	Impact of Dropout	11
7.4	Overfitting & Underfitting	13
7.5	KAN neurons compatibility with existing architectures	13
7.6	Backpropagation performance	13
7.7	Summary of Evaluation	14
7.8	Applications	14
8	Limitations	15
8.1	Strengths of the Approach	15
8.2	Pitfalls	15
9	Future Work	15
10	Conclusion	16
11	Acknowledgements	16
12	APPENDIX: Full Model Overview	20

Evaluation of Kolmogorov-Arnold Networks in Natural Language Processing: A Case Study in Sentiment Analysis

Kaloyan Tsankov

August 30, 2024

Abstract

This project evaluates the suitability of the recently proposed Kolmogorov-Arnold Neural Networks (KANs) for NLP tasks by conducting a case study in Sentiment Analysis on the IMDB dataset. KANs are a novel neural network architecture that replaces traditional linear weight layers with learnable activation functions defined by B-splines. This allows KANs to model complex, non-linear relationships using fewer parameters than standard multi-layer perceptrons. The report investigates whether the unique properties of KANs can provide an efficient yet accurate solution for the natural language processing domain. The overall architecture applied standard techniques, such as preprocessing, parallel CNNs and multilevel KANs, on the text reviews and evaluated its sentiment performance. It achieved 89.5% accuracy on the test set, besting the baseline CNN/LSTM models. Importantly, KANs offer potential future interpretability gains, making them a promising avenue for further research and development.

1 Introduction

Natural language processing (NLP) tasks like sentiment analysis are critical for understanding and analysing text data. From monitoring brand reputations to enabling intelligent stock market predictions, accurate sentiment analysis allows users to interpret the emotional tenor and subjective opinions over large amounts of data by reducing the manual intensity of the task. However, achieving high accuracy with traditional neural network models often requires large and complex architectures, resulting in extensive computational requirements for training and inference, as well as interpretability, limiting such models' scalability and real-time deployability. On the other hand, the recently proposed Kolmogorov-Arnold Neural Networks (KANs) offer an intriguing alternative that could address some of the said challenges with substantially lower computational cost. However, the authors of the original paper failed to test their approach on different than toy dataset examples.

2 Motivation

The recently introduced Kolmogorov-Arnold Neural Network is a strong and popular competitor of the traditional Multilayer perceptions. That is on paper. Despite the main article providing a multitude of examples, most of those are fitted with Toy Datasets and do not fully reflect the real-world scenario. Naturally, at the time of the writing, the several months-old novel approach also attracted a lot of criticism, rejecting a significant portion of the author's claims [6]. Even though not all of those reviews are research articles, their authors are part of the academic community. As a result, this project aims to realistically test KANs by harnessing their power against the widespread domain of Natural Language Processing. Some believe that KANs might be superior to MLPs by better handling complex, non-linear relationships and doing it with fewer network parameters. These qualities could make them a suitable and promising candidate for the domain. In the field of NLP, Large Language Models (LLMs) are one of the main topics for researchers. However, making these models better, more interpretable, smaller, and faster is imperative to provide even more general adaptation and wider adoption of their work. KANs might be well-fit for the challenge, as they require fewer parameters and have the potential to offer a more robust and explainable alternative, with performance gains by delegating pattern learning across fewer neurons. This project explicitly tests KANs for Sentiment Analysis. One might argue that if KANs

demonstrate even standard learning results in this context, it will justify further exploring their application to more complex tasks such as language models.

3 Literature Review

3.1 Trends in Sentiment Analysis

A quick survey of the top performers across Sentiment Analysis benchmarks reveals the overwhelming dominance of autoencoders and LLMs. However, there are increasing concerns about the sustainability of such tactics due to the heavy power consumption during the training and even inference phases.

This resonates with Csanády et al. [3], who expressed their concerns about LLM usage costs, especially for standard NLP tasks. Thus, they proposed LlamBERT - a hybrid unsupervised architecture, combining LLMs to annotate small unlabelled datasets, with a then fine-tuned autoencoder classifier like BERT. Nonetheless, merely running Llama2 (7B parameters) required a rather daunting computational power - NVIDIA A100 80GB VRAM GPU.

XLNet, yet another top achiever, could be summarised as an autoregressive pre-training of a slightly modified BERT to incorporate elements of TransformerXL. [20]

Similarly, Xie et al. [19] proposed Unsupervised data augmentation for consistency training (UDA). The authors declared, that the quality of the injected noise in the data is just as vital as the neural network itself. By leveraging Back-Translation, noise injection, word replacement, as well as other advanced Data Augmentation techniques in UDA, a fine-tuned BERT needed only twenty labelled examples to achieve the respectable 95,8% accuracy on the IMDB dataset. While impressive, this approach still requires many iterations as it effectively labels new samples with each epoch. It is effectively a type of semi-supervised training (self-training), combined with data augmentation policies. Those mentioned, as well as a large portion of the research on Sentiment Analysis, have focused primarily on those three components: Data augmentation, Autoencoders and LLMs, with occasional hints on energy efficiency and reduced computational overhead. [16, 21, 14]

3.2 Data Augmentation

As seen in the past decade, data preprocessing and augmentation have become critical for language-related tasks, especially for small domains or niche tasks where publicly available data is either scarce or expensive to label. Opinion mining, as a subset of NLP, also presents those symptoms, as real-world applications do not have the comfort of working with the carefully scraped and class-balanced IMDB dataset. In most scenarios, one has to work with minimal samples, even less labelled, whilst maintaining reasonable model performance and accuracy. Therefore, it is paramount not only to employ data augmentation, but also to select the best-fitting technique. The most notable methods could be categorised as random operations, back-translations, and LLM-based.

Early works in this area focused primarily on random operations [18]. In simple terms, the algorithms accept tokenised sentence data. Then, for each token, there is an equal chance to replace it with a synonym (synonym replacement (SR)), delete it entirely (deletion RD), change its place in the sentence (swapping RS), or add a random word next to it (insertion RI). The downside to such an approach is its inaccuracy. Synonym replacement is context-dependent, whereas most synonym checks are dictionary-based. In the best scenario, sentences are Part-of-Speech (POS) tagged to match synonym replacement by their POS type. As for rest, predominantly random insertions and deletions could introduce too much noise and completely change the overall sentence sentiment.

In a similar way, back-translations generate additional training samples by translating the sentence to a different language, which then gets translated back to the original. This creates slight variations of the original sentence without changing its context.

Original Sentence

Under the glow of the bioluminescent forest, the ancient tree whispered the secrets of the universe to the curious fireflies.

Japanese back-translation

Beneath the glow of the bioluminescent forest, ancient trees whispered secrets of the universe to inquisitive fireflies.

Nonetheless, it has its weaknesses due to the requirement of external resources (e.g. translation engines), further complicating the process and straining the available resources, whether financial for cloud solutions or computational in local environments.

That is equally true for LLM-based approaches, where a Language Model is either generative to create synthetic data or mask-based. The latter hides a token(s) from the original sentence, to generate a context-aware sentence variation.

Masked Text

Under the _ of the bioluminescent forest, the _ tree whispered the secrets of the universe to the _ fireflies.

Predicted Text

Under the *canopy* of the bioluminescent forest, the *ancient* tree whispered the secrets of the universe to the *glowing* fireflies.

Unfortunately, it is as effective as it is a costly solution, which would be justified only if the whole pipeline harnessed the power of Language Models. In those cases, without getting into much detail, one would ideally employ a pre-trained model or use API solutions like ChatGPT. The latter would produce additional expenses and increase future risks if the specified model version gets deprecated or the service shuts down. This is a severe security and contingency risk for companies. Moreover, there is an increasing concern of the impact of Large Language Models on the environment [5].

ChatGPT API Pricing		
	GPT-4o	GPT-4o mini
1M input tokens	\$5.00	\$0.150
1M output tokens	\$15.00	\$0.600
IMDB single pass (256 tokens x 40K)	~\$53.00	~\$1.61
Mask N = 3		

3.3 Convolutional Neural Networks for Sentiment Analysis

As this study primarily aims to evaluate KANs, their properties, and importantly computational effectiveness, it would be unwise to approach a sophisticated solutions like integrating KANs with autoencoders. However, a reasonably adequate performance could be found in simpler architectures like CNNs. Undoubtedly, CNNs work well with spatial data and are primarily known for their significant exploitation in Computer Vision. However, their capabilities are not unlimited, and while being great for extracting complex data patterns [11], they need to be combined with other architectures to show their potential. Notably, researchers, when tired of experiments with transformers and language models [20, 3], tackle sentiment analysis primarily with CNNs and a variation of Recurrent Neural Network, most commonly Long Short-Term Memory (LSTM) [15, 10, 8, 2]. Some of the main observations point that those architectures follow a certain pattern:

- The input is usually word-level, not character level; [15, 8, 9]
- Use word embeddings to create informationally rich feature space; [10]
- Parallel CNNs extract word n-gram information; [7, 9]
- Apply an RNN/LSTM over the CNN output; [2, 9]

However, LSTMs have one critical flaw—the information processing is sequential, meaning it relies on the previous steps. Consequently, both the training and inference stages are dramatically slowed down. Unfortunately, regular dense layers also fail the task due to their inability to capture long-distance word relationships. Another issue is their lack of understanding of complex linguistic tools, such as sarcasm, without the help of RNNs [9]. However, mitigating some of those impacts with KANs and their custom activation functions might be possible.

3.4 Kolmogorov-Arnold Networks (KANs)

The recently introduced Kolmogorov-Arnold Networks (KANs) have been a captivating force among the scientific AI community. Inspired by the Kolmogorov-Arnold representation theorem, the novel approach completely deviates from the dominating Multilayer-Perceptron (MLP) networks [13]. The extensive paper introduces the removal of linear weights by replacing them with learnable activation functions over the network’s edges. Those activation functions are primarily controlled by a configurable amount of B-splines, allowing the emergence of intricate functions compared to their primitive MLP counterparts like the sigmoid, ReLU or tanh activation functions. One of the primary advantages of KANs is their ability to fit with substantially fewer parameters than traditional neural networks. This is crucial for the new age of Natural Language Processing (NLP), as new reports express concerns about the energy consumption of large language models (LLMs) not only during training but also during inference stages [5]. Those networks’ efficiency and unique architecture allow more aggressive pruning, post-training fine-tuning, and potentially lower-cost scalability, all of which are valuable assets for deployment on computationally limited devices. While the authors primarily discuss the application of KANs in scientific tasks, the principles should be extendable to sentiment analysis. The ability to parse and model complex relationships within text data might lead to more accurate and insightful sentiment predictions without needing LLMs. Furthermore, the local plasticity of KANs helps mitigate issues like catastrophic forgetting, ensuring that the model retains previously learned sentiments while adapting to new data. Despite the likelihood that more primitive architectures, such as the one described in this report, will be insufficient for achieving state-of-the-art performance, the potential of KANs in NLP could be significant. A particularly intriguing results were achieved when KANs were integrated with other state-of-the-art architectures. For instance, combining KANs with transformer models (coined as "kansformers" in the unofficial GitHub implementation) try to leverage the strengths of both frameworks, enhancing the ability to process and analyse natural language efficiently. [4] Naturally, the bolder the claims, the more stringent the evaluation studies should be.

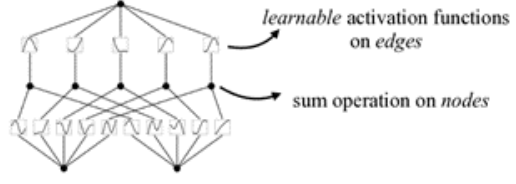


Fig 1: KAN: Activation Functions

4 Methodology

4.1 The Data

As sentiment analysis is a challenging task by itself, the project does not utilise scraping techniques but instead relies on the popular IMDB Movie Review dataset. It contains raw review information, including HTML tags and a Boolean representation of the positive or negative sentiment. Similar to spam detection and other tasks with likely under-represented classes, the algorithm required balance checks of the dataset. Fortunately, positive and negative classes were split equally, meaning there was no class bias, and no further action was needed.

Natural language processing has always been constrained to context windows. All variable-length inputs needed to receive a uniform and equal shape. Analysis of the dataset revealed that the mean unprocessed review consisted of approximately 230 whitespace separable tokens. This later led to input truncation and padding to 256 tokens. One would argue that the network lost too much context when dealing with long data, but as one can see, such long reviews are an outlier in this domain. In other words, it would simply be cost-ineffective to process 300+ context windows length. In terms of preprocessing, the dataset underwent basic regex filtering to remove HTML remnants and non-alphanumeric values (e.g. HTML tags). Furthermore, with the help of NLTK, stopwords were scrapped as they offer little to no linguistic benefits while taking up valuable token space. To further decrease the unique tokens, all characters had to be lowercase.

Next, the reviews were tokenised via the more advanced GPT2Tokenizer, which allows out-of-training vocabulary to be represented as multiple tokens. For example, “chartreuse” would result in “chart,” “re,” and “use” tokens. Last but not least, an 80-20 training-testing split with a batch size of 64 was applied to the dataset, marking the end of the preprocessing stage.

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

Fig 2: The IMDB dataset

Positive review length	
count	25000.000000
mean	232.849320
std	177.497046
min	10.000000
25%	125.000000
50%	172.000000
75%	284.000000
max	2470.000000
Negative review length	
count	25000.000000
mean	229.464560
std	164.947795
min	4.000000
25%	128.000000
50%	174.000000
75%	278.000000
max	1522.000000

Fig 3: Data Length

review	clean
One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are right, as this is exactly what happened with me. The first thing that struck me...	one reviewers mentioned watching 1 oz episode youll hooked right exactly happened methe first thing struck oz brutality unflinching scenes violence set right word go trust show faint hearted timid...

Fig 4: Processed Data

4.2 Easy Data Augmentation

To avoid the use of computationally expensive architectures, or external/cloud resources, the selected data augmentation type was random operations. Although it was possible to re-implement the approaches described above, it would fall outside of the scope of this project. Hence, this step adopts the outstanding Easy Data Augmentation (EDA) solution by Wei and Zou [18]. EDA combines synonym replacement, random insertion, random swap, and random deletion given an initially configured probability P . A variety of settings were tested and documented in the Experimental Section of this report.

4.3 The Neural Network

The network consists of input + embeddings, three parallel convolutional layers with dropout 25%, concatenated into a flattened tensor, three additional sequential convolutional layers with dropout 50%, and four KAN layers of sizes 256, 64, and 8 with an output of size two. The convolutional layers apply filters to extract sentiment patterns from the embeddings over n-gram words from the review, where n=1, 3, 5 respectively. Then, a concatenation layer joins and flattens the parallel output, feeding it into the sequential part of the network and the KAN layers.

As previously mentioned, Kolmogorov-Arnold Networks are pretty different from the standard MLP, given that the activation happens not inside the neuron but on their edges. Additionally, this activation function is learnable and unique for each edge. Mathematically, the Kolmogorov-Arnold representation for any multivariate continuous function f is expressed as [13]:

$$f(\mathbf{x}) = f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \phi_{q,p}(x_p) \right)$$

Put simply, it is the finite composition of continuous functions of a single variable and their addition. Delving into the KAN layers, each edge acts as the inner sum operator $\Phi_q(\sum_{p=1}^n \phi_{q,p}(x_p))$ while the neurons perform sum over all edges. The authors use B-splines, which are almost identical to the famous Bezier curves, to enable stable backpropagation over the polynomial edge functions. B-splines are highly accurate at low dimensions, and piecewise polynomial curves are continuous and differentiable, which is necessary for backpropagation. They allow for local adjustments through modifications of single control points without disturbing the global shape of the curve.

The main downside and bottleneck of this specific implementation lies in its backpropagation speed. As mentioned above, Li [12] declare that it is safe to approximate the B-Spline basis in KANs with Gaussian Radial Basis Function approximators. Although early versions of this project reached out to the original KAN method, the final research adopted Li [12] solution to decrease the experiment runtimes. Further details on the referenced methodology can be found in “Kolmogorov-Arnold Networks are Radial Basis Function Networks”. Full network overview available in the APENDIX pages.

5 Implementation

5.1 Hardware

The project was implemented and tested on the following hardware:

System Information	
Item	Value
System Model	Alienware x17 R1
OS Name	Microsoft Windows 11 Home
Version	10.0.22635 Build 22635
Processor	11th Gen Intel(R) Core(TM) i9-11980HK @ 2.60GHz, 3302 Mhz, 8 Core(s), 16 Logical Processor(s)
Installed Physical Memory (RAM)	48.0 GB
Graphics Processing Unit (GPU)	NVIDIA RTX3080 Mobile
Graphics Processing Unit (GPU) Memory	16.0 GB

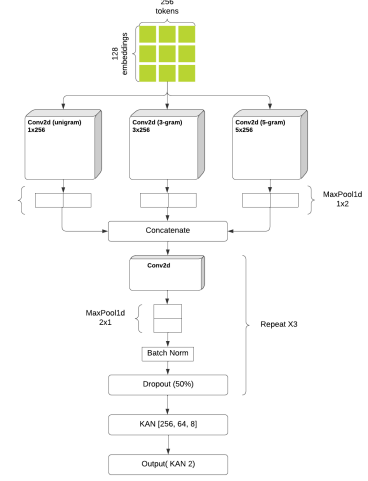


Fig 5: Model Architecture

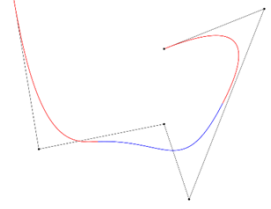


Fig 6: B-Splines

5.2 Technical Approach

This research was fully implemented with the help of five Python scripts. ¹

1. **sakan.py**

The file combines all required modules to execute the Sentiment Analysis pipeline.

It starts by loading and preprocessing the data (`data_preprocess.py`), then initialises and trains the model (`model.py` & `trainer.py`) and finishes by displaying the model's training history (`visualiser.py`) and saving it to disk.

2. **data_preprocess.py**

It contains the `DataProcessor` class, responsible for loading, cleaning and preparing the data for the model.

The labels are converted into a numerical form (1 for positive and 0 - negative sentiment). Then the reviews are converted to lowercase and cleaned of HTML tags with regex. The NLTK library removes stopwords. The cleaned data is lemmatised with `WordNetLemmatizer` and finally tokenised with the `GPT2Tokenizer`.

3. **model.py**

Stores the model architectures.

4. **trainer.py**

Loads the data into tensors and trains the model.

5. **visualiser.py**

Uses `Graphviz` to create a visual model of the architecture.

Plots the model history.

Libraries used in the implementation:

- **torch**: Creation (via Torch API) and training of the neural networks.
- **pandas**
 - Used for data manipulation and analysis
 - Loads and processes CSV files
- **nlk**
 - Natural Language Toolkit for text processing
 - Used for stop words removal and lemmatisation
- **re**
 - Provides support for regular expressions
 - Used for text cleaning in preprocessing
- **transformers**
 - Hugging Face's library for NLP models
 - Used for GPT-2 tokeniser
- **tqdm**: Progress feedback during training and data preprocessing.
- **matplotlib**: Creates accuracy and loss plots.
- **torchviz**: Used to create a visual representation of the model architecture.
- **os**: Used as a fail-safe for setting environment variables and file path operations.
- **fastkan**
 - Custom library for Kolmogorov-Arnold Networks (KAN)
 - Used in the custom model as a replacement of the fully-connected layer

¹Full implementation with archived experiments available at: <https://github.com/TheCyberBug/SAKAN>

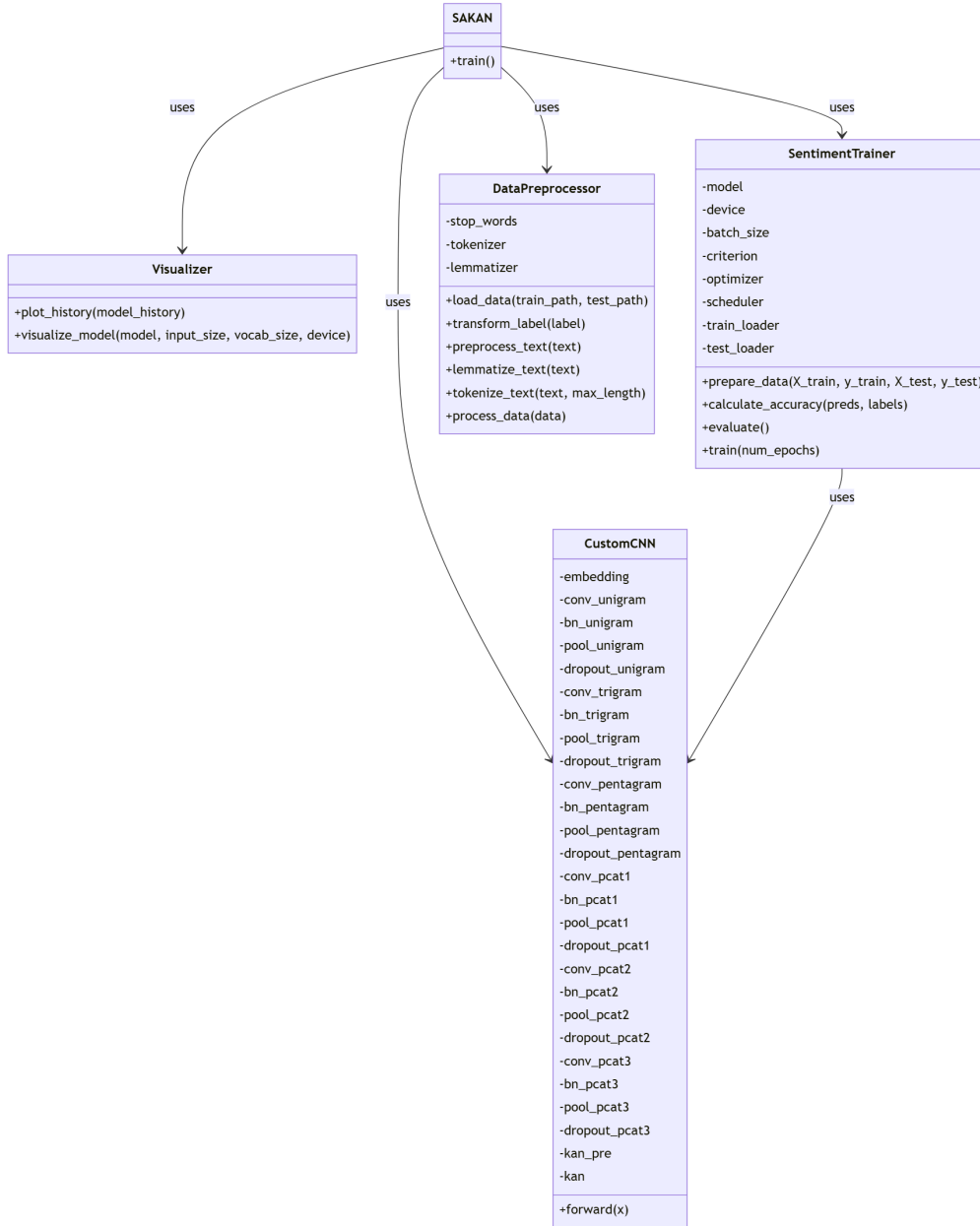


Fig 7: UML of the Implementation

6 Challenges

A relative challenge was to tackle model speed and efficiency, as early implementations used the official pykan library provided with the KAN paper. However, upon further investigation, a more efficient candidate was identified: the FastKAN library. [12]

Furthermore, tokenisation was an issue as the IMDB dataset contains irregularities, such as missing whitespaces between words. However, compared to the standard Keras tokeniser, the pre-trained GPT2Tokenizer better handled complex words and missing spaces by splitting them apart, resulting in a more robust output and increased accuracy in the final versions.

Finally, keeping the model from overfitting was much more specific compared to MLPs. This was presumably due to the nature of B-splines and their high expressiveness. Whilst dropout positively impacted, the data augmentation applied deepened the issue, and was completely disabled. Reducing the number of B-spline parameters improved training and overall robustness.

7 Experimental Results and Evaluation

7.1 Task evaluation

The experimental case study yielded an acceptable performance given the deployed evaluation stages. On the original dataset, training times were less than two and a half minutes per epoch while producing stable $89.5 \pm 0.18\%$ accuracy on the test set over five consecutive runs. Despite the accuracy being somewhat low compared to the state-of-the-art results, it must be noted that this was a much more simplistic architecture compared to the top performers. Excluding transformers, pure CNN+MLP and CNN+LSTM publications featured 88.1% and 88.9% accuracy, which were bested by the current implementation. [2]

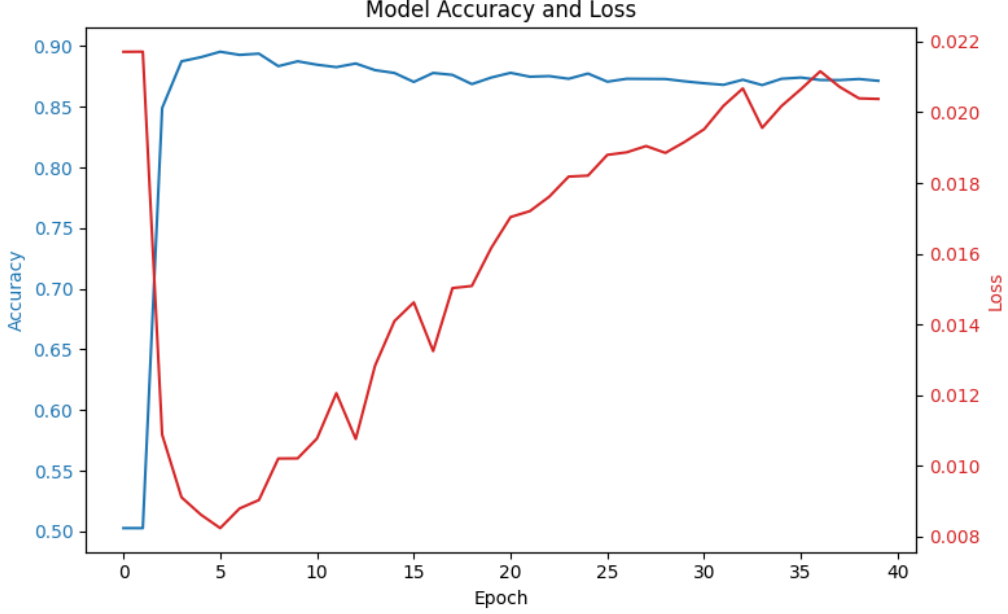


Fig 8: Model History

One might notice that the model converged at an unexpectedly quick pace but showed tendencies towards overfitting despite the efforts to prevent it from doing so. Further investigation is required, but intriguingly, the overall validation accuracy stays stable even with the increase in average loss.

7.2 Impact of EDA

The data augmentation technique was not utilised until the most capable architecture was identified. However, to much of a surprise, the Easy Data Augmentation had a negative impact on the overall performance of the network. This decrease might be due to the tendency of KANs to overfit. In an attempt to confront such behaviour, EDA was retested with various configurations. The first of those configurations increased the number of augmented output sentences N and $SR||RS||RI||RD$ operations per given original sequence. This was supposed to increase sentence variability and decrease overfitting, but it had the opposite effect - the network overfit in fewer epochs than before. Due to speculated impact of a one or more operations being the root of the accuracy drop, individual testing of isolated augmentation operations was initiated. In summary, the overall prediction quality worsened even with a single augmented sentence. There might be various reasons for such a decrease, but most likely, the simplistic nature of those data augmentation policies could not match the strong learning capabilities of KANs, essentially destroying the model's generalisation.

7.3 Impact of Dropout

Implementing dropout layers is common for convolutions-based neural networks. The goal is to slow down and ultimately prevent overfitting by inducing neuronal independence and promoting broader feature importance. Likewise were the reasons behind its introduction in this project, but

EDA Impact						
N - Augmented	1	4	1	1	1	1
SR%	10%	20%	20%	0%	0%	0%
RS%	10%	20%	0%	20%	0%	0%
RI%	10%	20%	0%	0%	20%	0%
RD%	10%	20%	0%	0%	0%	20%
Impact	NEG	NEG	NEG	NEG	NEG	NEG

their effectiveness proved to be less than expected. There was a dropout layer between each stack of the convolutional layers with various settings tested between experiments:

1. No dropout
The network overfitted with constant oscillations in the learning curve from the very start.
Experimental conclusions: Dropout introduction was necessary.
2. Dropout 25% for early pre-concatenated layers and 25% for the rest.
The network overfitted with constant oscillations and had low maximum accuracy, but a wide slope-shaped curvature was spotted.
Experimental conclusions: Dropout ratio should be increased in post-concatenation layers.
3. Dropout 25% for early pre-concatenated layers and 40% for the rest.
The network quickly overfitted, but the learning curve had a more defined U-shape with fewer oscillations.
Experimental conclusions: Dropout ratio should be increased further in post-concatenation layers.
4. Dropout 25% for early pre-concatenated layers and 50% for the rest.
The network quickly reached maximum performance but overfitted in subsequent epochs.
Clear U-shaped learning curve with almost no oscillation over it.
Experimental conclusions: The ratio should be increased in the early layers.
Note: This configuration was chosen as final.
5. Dropout 50% for early pre-concatenated layers, 50% mid layers, and 60% for the final.
Slight reduction in performance, but it kept overfitting in later stages.
Experimental conclusions: The lesser ratio was optimal for the early layers. The higher it gets, the less relevant patterns are passed onto subsequent layers and clear patterns could not be picked up by the network.
6. Dropout 25% for early pre-concatenated layers, 50% mid layers, and 60% for the final.
Slightly reduced accuracy. Clear U-shaped curvature. Overfits in later epochs.
Experimental conclusions: Dropout could stop overfitting but at the cost of accuracy.

Overall, dropout had less of an impact on accuracy or over-fitting and more on the time required for progression of the learning curve and its stability. A surprising finding for the network was that it preserved its relative accuracy once it reached the maximum, despite the increases in validation loss with each epoch. Once the network picks up a slight data pattern (reaches approximately 60+%), its performance skyrockets up to 80% for as little as two epochs. The main observations of the effectiveness of the dropout layer were in early training, where the higher the dropout ratio, the longer the network required to achieve that minimal threshold of 60%.

One would rightly speculate that such erratic model behaviour clearly indicates a too complex network architecture; thus, the KAN layer sizes should have been reduced. This possibility was experimentally tested, but optimal performance was only available in the described setup.

Another likely possibility was that the network was not deep enough, as the hidden input size towards the KAN layers bordered forty thousand. However, as explained in the network architecture section, deeper networks were implemented but had no additional gains. Regardless, further research is required as the computational requirements for a speculated optimal depth far exceeded the hardware limitations.

Due to the highly-sensitive inter-neuronal connections, the observed behaviour could be interpreted as a strong long-term memory indication. Nonetheless, further research is required to confirm such a theory.

7.4 Overfitting & Underfitting

As mentioned above, the network expressed non-standard properties, where the slowest part of the training was the initial jump from 50% to 60%. Afterwards the optimal for the model performance was almost every time reached within four to six epochs.

Based on tiny changes to the initial network configuration, the resulting behaviour was either quick overfitting, or sub-optimal underfitted model. Balancing it into an optimal shape of the learning curve deemed to be a rather complex target with few options for execution, given the limited hardware resources at hand.

The initial learning rate was the most influential hyperparameter, single-handedly allowing better control over the network’s behaviour. However, similarly to the unexpected dropout behaviour, it did not allow fine-grained changes, but rather erratic and sudden thresholds between over and underfitting. Generally, the network favoured a higher learning rate than the typical, which resulted in training instability (oscillations). Attempts to reduce it had little impact on the number of oscillations, but prolonging them in time. After a certain threshold, even the slightest reduction in the learning rate would eradicate them, but at the cost of leaving the model into a local minima (underfitting). Thankfully, dropout resolved the oscillation.

The authors of the original paper confirm this tendency to overfitting. Their suggestion is to carefully configure the KAN’s parameters in the following order of importance:

- Number of grids
- Layer width
- Number of layers

None of the suggestions improved the most accurate model, suggesting it had reached its architectural limit. Interestingly, these correlations between all previously discussed findings might hint that the general application of this network type would lie better in line with data-intensive tasks with large training datasets. This way, it would be reasonably probable to assume a reduction of overfitting, whilst deeply engraving long-term feature patterns during training. More details on those assumptions and hypothesis can be found in the applications section of this report.

7.5 KAN neurons compatibility with existing architectures

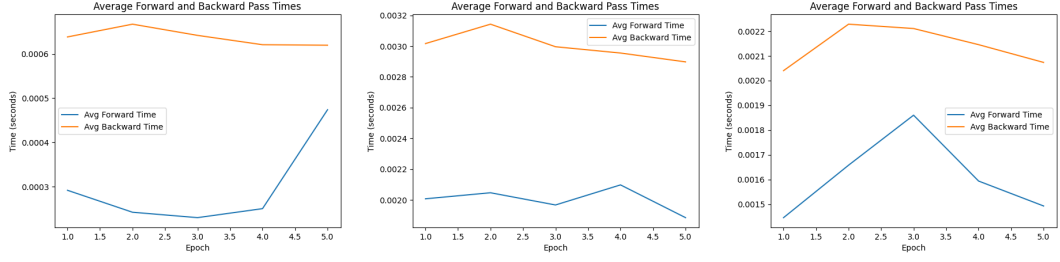
Kolmogorov-Arnold Networks are too architecturally and operationally different from their MLP counterparts. To a large regard, the user’s accumulated MLP knowledge and experience are non-applicable for KANs.

The single most crucial tip for future research and development would be to start with a small network and gradually increase the hyperparameters for grids per neuron, layer width and number of layers. Based on KAN libraries at the time of writing this article (PyKAN, efficientkan and FastKAN), typical multi-layer KANs declaration is possible in a single Pytorch line. However, it would be wiser to split those layers apart and deploy unique grid sizes for the individual layer groups. For example, during integration with existing techniques, the bridge layer could start with a lower grid size, to allow a gradual conversion between the standard ML towards KAN layers by reduction in MLP induced neuronal noise. Speculatively, it might reduce overfitting to the MLP input and allow for down-the-line generalisation. The next layers could still use larger grid sizes as there it is expected for the network to efficiently recognise complex patterns.

Moreover, one should consider exploring the KAN-specific dropout proposal by Altarabichi [1] compared to the traditional dropout usage described in this paper. Unfortunately, it was not possible to test its effectiveness, as it was published after the cessation of the experimental stage of this article.

7.6 Backpropagation performance

A new experimental setup was implemented to evaluate forward and backpropagation times. The statistical data suggest a considerable performance drawback in the novel network, even with the RBF performance gains, compared to standard fully connected neuronal architectures. In the best of scenarios, KAN with equal to MLP $N_{neurons}$ was 5.6 times slower, whereas in the case of matching $N_{parameters}$, the drop was 4.3 times. Those results, describe a significant drawback of Kolmogorov-Arnold networks, especially concerning for data-intensive domains.



(a) MLP per input training times (b) KAN: Same N neurons as MLP (c) KAN: Approx. N param as MLP

Fig 9: Comparison of per input training times of MLP and KANs

Layer (type:depth-idx)	Output Shape	Param #	Layer (type:depth-idx)	Output Shape	Param #
MLPNet	[1, 10]	--	MLPNet	[1, 10]	--
└Flatten: 1-1	[1, 784]	--	└Flatten: 1-1	[1, 784]	--
└Linear: 1-2	[1, 64]	50,240	└Linear: 1-2	[1, 64]	50,240
└ReLU: 1-3	[1, 64]	--	└ReLU: 1-3	[1, 64]	--
└Linear: 1-4	[1, 10]	650	└Linear: 1-4	[1, 10]	650
Total params: 50,890			Total params: 50,890		
Trainable params: 50,890			Trainable params: 50,890		
Non-trainable params: 0			Non-trainable params: 0		
Total mult-adds (M): 0.05			Total mult-adds (M): 0.05		
Input size (MB): 0.00			Input size (MB): 0.00		
Forward/backward pass size (MB): 0.00			Forward/backward pass size (MB): 0.00		
Params size (MB): 0.20			Params size (MB): 0.20		
Estimated Total Size (MB): 0.21			Estimated Total Size (MB): 0.21		
Layer (type:depth-idx)	Output Shape	Param #	Layer (type:depth-idx)	Output Shape	Param #
KANNet	[1, 10]	--	KANNet	[1, 10]	--
└Flatten: 1-1	[1, 784]	--	└Flatten: 1-1	[1, 784]	--
└FastKAN: 1-2	[1, 10]	--	└FastKAN: 1-2	[1, 10]	--
└ModuleList: 2-1	--	--	└ModuleList: 2-1	--	--
└└FastKANLayer: 3-1	[1, 64]	453,224	└└FastKANLayer: 3-1	[1, 7]	50,975
└└FastKANLayer: 3-2	[1, 10]	5,906	└└FastKANLayer: 3-2	[1, 10]	662
Total params: 459,130			Total params: 51,637		
Trainable params: 459,114			Trainable params: 51,621		
Non-trainable params: 16			Non-trainable params: 16		
Total mult-adds (M): 0.46			Total mult-adds (M): 0.05		
Input size (MB): 0.00			Input size (MB): 0.00		
Forward/backward pass size (MB): 0.06			Forward/backward pass size (MB): 0.06		
Params size (MB): 1.84			Params size (MB): 0.21		
Estimated Total Size (MB): 1.90			Estimated Total Size (MB): 0.27		

(a) KAN: Same N neurons

(b) KAN: Approx. N parameters

Fig 10: Comparison Setup of KAN and the MLP counterpart

7.7 Summary of Evaluation

The Kolmogorov-Arnold networks have shown potential in NLP, especially due to their increased expressiveness, which could benefit the differentiation of the abstractions propagated by word senses. However, there is a price in the face of slower training times, which are crucial, especially for large networks like autoencoders. However, it is still unclear, whether the reduction in the total neurons will mitigate this drawback. Furthermore, they have niche application targets for tasks with an abundance of data, preferably combined with semi or unsupervised training. Yet, their most significant drawback lies in the distinction between working AI approaches and common practices.

7.8 Applications

The recently introduced network architecture exhibited properties that were in contrast with standard neural networks. Whilst it was modelled after MLPs and theoretically they are fully compatible and interchangeable, there are several caveats that prevent a straightforward process to exist. Firstly, the KAN-converted network requires tuning to the number of neurons, as they encode richer information for each synaptic connection. Then, the existing optimisation and performance-boosting techniques are either inapplicable or, at the very least, require additional alterations (e.g. dropout). Additionally, they are a computationally intensive and expensive approach that should be deployed mainly for feature-rich tasks. This is primarily to prevent overfitting. Even then, the slower backpropagation and forward pass would be detrimental to assessing the suitability for the given task. Overall, the findings of this study show several implications for KANs, but they

could still be a viable option in the NLP domain. One such application, might be for Language Models, where the transformer architecture has the most potential of benefitting from the rich pattern recognition and feature-capturing properties. Moreover, as they tend to use enormous corpora for training, speculatively, the network should not be able to overfit. Unfortunately, those potential gains, if not outstanding, could easily be overshadowed by the additional training time the KAN-Transformers would need.

On the other hand, the most suitable domains would be those where interpretability is paramount, such as medical research or business solutions that make decisions on people’s lives (e.g. loan applications, self-driving cars). In those cases, the implemented model would require less total neurons, potentially allowing the model to be directly interpretable.

As in the Sentiment Analysis’ context, as it achieved higher than the baseline performance of CNN+MLP model, the applications could vary from brand image monitoring to market research. However, due to its inherent drawbacks, it would be more favourable to use standard solutions where possible. A deviation of those cases might be sentiment analysis on product reviews for large companies, like Amazon, where there is an abundance of data and resources.

8 Limitations

8.1 Strengths of the Approach

Even though the architecture is experimental and in its early stages, it has reached the baseline performance requirements of the vanilla CNN / CNN + LSTM networks. This is an essential declaration of the network’s capabilities as a whole. Furthermore, the model’s main strength lies in its increased expressiveness due to the KAN’s operating nature. It allows for more interpretable, smaller-size implementations to achieve respectable performance and allows mid-range computationally able devices to operate the model locally.

8.2 Pitfalls

As a novel approach, normal intuition is not readily applicable here, and there are no current peer-reviewed publications to shine a better light on the optimal conditions for KANs to operate. Generally speaking, it is not a straightforward MLP replacement that can be easily adapted for any given task.

The learning curve is especially big for those who attempt to use the approach for the first time, as most would rely on transferable knowledge from previous projects. Unfortunately, there is a substantial gap between the intuitive solutions in standard networks and the special resolutions required by KANs. Some of the examples discussed previously were the dropout, as well as the unexceptionably high learning rate required, and the failure of data augmentation policies to positively affect performance. Crucially, one should be very cautious not to allow the network to over or underfit the data, as KANs, due to their much expressive B-splines, are prone to this issue.

In the context of the current case study, one should note that it is an early attempt to experimentally evaluate the suitability of the said approach in the field of NLP. Still, more optimisation steps, architectural redesigning, and better hardware are needed for a full evaluation before pronouncing a final verdict.

9 Future Work

The project utilised only the barebone functionalities of KANs. However, numerous improvements are planned for all network layers. To begin with, one of the key strengths of KANs is their use of B-splines and the possibility of finetuning their amount at different stages of the network. For example, one might define a four-point B-spline, train the model, and then fine-tune it by increasing the number of defined points and continuing training. This would speed up early training phases, reduce the risk of trapping the model into a local minima and increase the cost-effectiveness during training. Additionally, libraries are already experimenting with KAN-based convolutional layers, which should create a more expressive filtering stage. Furthermore, KANs could be better pruned than their MLP counterparts because the activation functions are more suited for the task and require less overall neuron connectivity. Combined with B-spline finetuning, this would likely create an even more robust and smaller architecture with minimal impact on performance. Finally, as one might have noticed, the current CNN layers operate over the whole embedding depth.

Future versions aim to experiment with less coarse convolutions to increase the model’s accuracy and expressiveness.

10 Conclusion

Overall, this study explored the novel Kolmogorov-Arnold Neural Network (KAN) architecture for sentiment analysis and achieved a base level of accuracy with limited and primitive architecture. While not state-of-the-art, these promising results demonstrate the potential of KANs if correctly utilised in the NLP domain. Thanks to their increased expressiveness and compact model size, KANs offer an appealing alternative that could help mitigate the growing demands of large language models. Future research is required to further boost performance by combining KANs with other architectures, advanced B-spline control, and model compression techniques. Although they exhibit numerous drawbacks, overall, KANs have shown initial competency for NLP, which merits further exploration of their applicability across a broader range of linguistic tasks.

11 Acknowledgements

I am deeply indebted to my tutor, Dr Brett Drury, whose steadfast guidance, insightful feedback, and scholarly mentorship have been instrumental in shaping the trajectory of this dissertation. Dr Drury’s profound expertise in Natural Language Processing and his unyielding commitment to academic excellence have left an indelible mark on my academic and personal growth. This research wouldn’t be possible without his meticulous review, constructive feedback, and thought-provoking discussions, which greatly enriched the development of the evaluation study of KANs for Sentiment Analysis.

I am immensely appreciative of the faculty members at Liverpool Hope University for fostering an intellectually stimulating environment that nurtured my curiosity and provided the resources essential for this study. Their collective wisdom has been pivotal in my academic journey.

Special mention goes to my supervisor for his generous assistance in refining my academic writing. His guidance and insights have significantly elevated the clarity and coherence of my works throughout the years.

I extend my sincere thanks to my peers and colleagues for their engaging discussions, camaraderie, and unwavering support throughout the course of this research.

Finally, I am profoundly grateful to my family and friends for their enduring encouragement and belief in my capabilities. Their unwavering support has been a wellspring of motivation and resilience.

References

- [1] Mohammed Ghaith Altarabichi. *DropKAN: Regularizing KANs by masking post-activations*. 2024. arXiv: [2407.13044](https://arxiv.org/abs/2407.13044) [cs.LG]. URL: <https://arxiv.org/abs/2407.13044>.
- [2] Jose Camacho-Collados and Mohammad Taher Pilehvar. “On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis”. In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Ed. by Tal Linzen, Grzegorz Chrupala, and Afra Alishahi. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 40–46. DOI: [10.18653/v1/W18-5406](https://doi.org/10.18653/v1/W18-5406). URL: <https://aclanthology.org/W18-5406>.
- [3] Bálint Csanády et al. “LlamBERT: Large-scale low-cost data annotation in NLP”. In: (2024). arXiv: [2403.15938](https://arxiv.org/abs/2403.15938) [cs.CL]. URL: <https://arxiv.org/abs/2403.15938>.
- [4] Akaash Dash. *akaashdash/kansformers*. GitHub, May 2024. URL: <https://github.com/akaashdash/kansformers> (visited on 05/31/2024).
- [5] Charlotte Debus et al. “Reporting electricity consumption is essential for sustainable AI”. In: *Nature Machine Intelligence* 5 (Nov. 2023), pp. 1176–1178. DOI: [10.1038/s42256-023-00750-1](https://doi.org/10.1038/s42256-023-00750-1). URL: <https://www.nature.com/articles/s42256-023-00750-1> (visited on 01/09/2024).
- [6] Vikas Dhiman. *KAN: Kolmogorov-Arnold Networks: A review*. 2024. URL: https://vikasdhiman.info/reviews/KAN_a_review.pdf (visited on 05/31/2024).
- [7] Yazhou Hao et al. “Improving Chinese Sentiment Analysis via Segmentation-Based Representation Using Parallel CNN”. In: (Nov. 2017), pp. 668–680. DOI: [10.1007/978-3-319-69179-4_47](https://doi.org/10.1007/978-3-319-69179-4_47). (Visited on 05/26/2023).
- [8] Jeremy Howard and Sebastian Ruder. “Universal Language Model Fine-tuning for Text Classification”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2018). DOI: [10.18653/v1/p18-1031](https://doi.org/10.18653/v1/p18-1031). URL: <https://aclweb.org/anthology/P18-1031> (visited on 10/18/2019).
- [9] Deepak Jain, Akshi Kumar, and Geetanjali Garg. “Sarcasm detection in mash-up language using soft-attention based bi-directional LSTM and feature-rich CNN”. In: *Applied Soft Computing* 91 (June 2020), p. 106198. DOI: [10.1016/j.asoc.2020.106198](https://doi.org/10.1016/j.asoc.2020.106198). URL: <https://www.sciencedirect.com/science/article/abs/pii/S1568494620301381> (visited on 04/04/2020).
- [10] Rie Johnson and Tong Zhang. “Supervised and semi-supervised text categorization using LSTM for region embeddings”. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML’16. New York, NY, USA: JMLR.org, 2016, pp. 526–534.
- [11] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86 (1998), pp. 2278–2324. DOI: [10.1109/5.726791](https://doi.org/10.1109/5.726791).
- [12] Ziyao Li. “Kolmogorov-Arnold Networks are Radial Basis Function Networks”. In: (2024). arXiv: [2405.06721](https://arxiv.org/abs/2405.06721) [cs.LG].
- [13] Ziming Liu et al. *KAN: Kolmogorov-Arnold Networks*. arXiv.org, 2024. URL: <https://arxiv.org/abs/2404.19756>.
- [14] Gibson Nkhata, U Anjun, and Justin Zhan. “Sentiment analysis of movie reviews using bert”. In: *The Fifteenth International Conference on Information, Process, and Knowledge Management, eKNOW23*. 2023.
- [15] Devendra Singh Sachan, Manzil Zaheer, and Ruslan Salakhutdinov. “Revisiting LSTM Networks for Semi-Supervised Text Classification via Mixed Objective Function”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (July 2019), pp. 6940–6948. DOI: [10.1609/aaai.v33i01.33016940](https://doi.org/10.1609/aaai.v33i01.33016940). (Visited on 02/28/2020).
- [16] Chi Sun et al. “How to Fine-Tune BERT for Text Classification?” In: *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings*. Kunming, China: Springer-Verlag, 2019, pp. 194–206. ISBN: 978-3-030-32380-6. DOI: [10.1007/978-3-030-32381-3_16](https://doi.org/10.1007/978-3-030-32381-3_16). URL: https://doi.org/10.1007/978-3-030-32381-3_16.
- [17] Kaloyan Tsankov. *Sentiment Analysis with KANs*. Coursework submitted for COMM013AZ2023/4 - Artificial Intelligence (2023/4). 2023.

- [18] Jason Wei and Kai Zou. “EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro Inui et al. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6382–6388. DOI: [10.18653/v1/D19-1670](https://doi.org/10.18653/v1/D19-1670). URL: <https://aclanthology.org/D19-1670>.
- [19] Qizhe Xie et al. “Unsupervised data augmentation for consistency training”. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS ’20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.
- [20] Zhilin Yang et al. “XLNet: generalized autoregressive pretraining for language understanding”. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.
- [21] Ye Zihao et al. “BP-Transformer: Modelling Long-Range Context via Binary Partitioning”. In: (Nov. 2019).

Glossary

B-spline A type of mathematical curve used in computer graphics and computer-aided design. In KANs, B-splines are used to define learnable activation functions. 3

Back-Translation for a given sentence sequence S , translates the sentence in another language. Then translates it back into the original language. Relies on the concept, that during translations, words are substituted by in-context synonyms.. 4

BERT a pretrained auto-encoder classifier. 4

CNN Convolutional Neural Network, a class of deep neural networks most commonly applied to analysing visual imagery. It applies element-wise multiplication of filters over the data. The resulting matrix is summed into a single element. Based on the mathematical concept of convolutions.. 3

Data Augmentation A suite of techniques to diversify and enlarge a dataset. Usually applied to small datasets, where the labelled samples are not nearly enough for full AI/ML training.. 4

IMDB Internet Movie Database, a dataset used for sentiment analysis tasks, containing movie reviews and their associated sentiments - either positive or negative. 3, 4

KAN Kolmogorov-Arnold Network, a novel neural network architecture that replaces traditional linear weight layers with learnable activation functions defined by B-splines. 3

Llama2 Large Language Model by Meta. 4

LlamBERT Hybrid unsupervised architecture, combining LLMs to annotate small unlabelled datasets, with fine-tuned autoencoder classifiers like BERT.. 4

LLM Large Language Model, an AI model trained on vast amounts of text data to generate human-like text and perform various language tasks. 3, 4

LSTM Long Short-Term Memory, a type of recurrent neural network capable of learning long-term dependencies. 3

MLP Multilayer Perceptron, a class of feedforward artificial neural network. 3

NLP Natural Language Processing, a field of artificial intelligence that focuses on the interaction between computers and humans using natural language. 3, 4

Toy Dataset small and simple artificial collection of data for demonstrating concepts, algorithms, or techniques in machine learning. They help to unravel the underlying patterns in prototype models without the computational burden of large, real-world datasets.. 3

12 APPENDIX: Full Model Overview

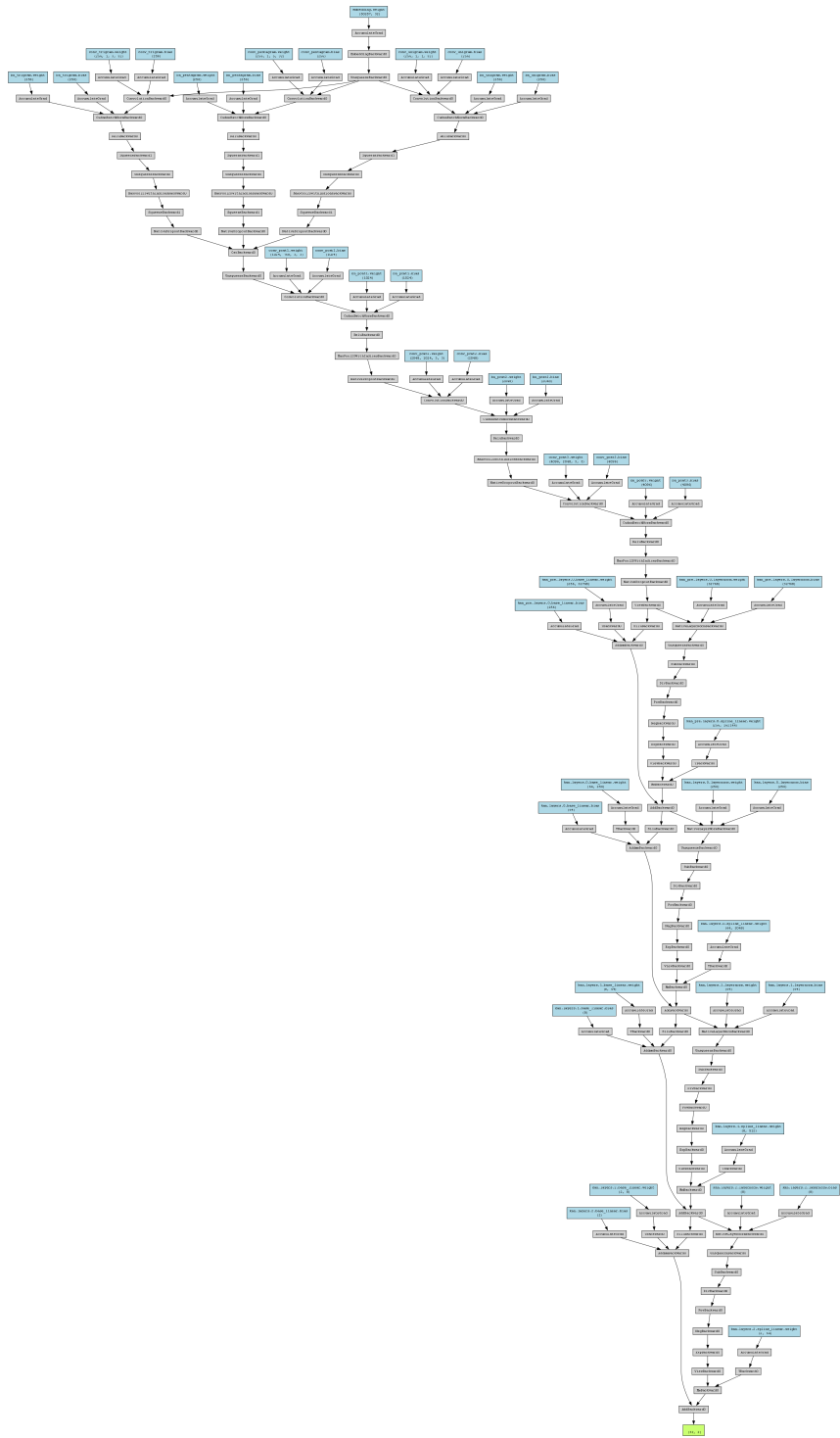


Fig 11: Full Model Overview