Start Simple: Progressive Difficulty Multitask Learning

Yunfei Luo
University of California, San Diego
yul268@ucsd.edu

Yuyang Liu
Yale University
yuyang.liu@yale.edu

Rukai Cai
University of Massachusetts, Amherst
rukaicai@umass.edu

Tauhidur Rahman
University of California, San Diego
trahman@ucsd.edu

Abstract

The opaque nature of neural networks, often described as black boxes, poses significant challenges in understanding their learning mechanisms, which limit our ability to fully optimize and trust these models. Inspired by how humans learn, this paper proposes a novel neural network training strategy that employs multitask learning with progressive difficulty subtasks, which we believe can potentially shed light on the internal learning mechanisms of neural networks. We implemented this strategy across a range of NLP tasks, data sets, and neural network architectures and observed notable improvements in model performance. This suggests that neural networks may be able to extract common features and internalize shared representations across similar subtasks that differ in their difficulty. Analyzing this strategy could lead us to more interpretable and robust neural networks, enhancing both their performance and our understanding of their nature.

1 Introduction

How do neural networks learn? This question remains a complex and intriguing area in the field. Despite the substantial advancements in the application and performance of neural networks nowadays, a comprehensive understanding of the logical connection between their internal configurations and external behaviors is still developing (a.o.: Wildberger, 1994; Lipton, 2018; Rudin et al., 2022).

What we might want to seek intuition from, however, is how humans learn. One of the key observations in this heavily researched field (e.g., Lovett et al., 2023) is that people's learning process can be facilitated by starting from understanding simple notions or from solving toy problems. Inspired by this observation, we propose a multitask learning (MTL) strategy (Caruana, 1997) that trains a neural network using subtasks of progressive difficulty. We apply this strategy to train neural networks across different NLP tasks: sentiment analysis, text classification, unit segmentation, and syllogistic reasoning. We also experiment with training different types of neural networks using this strategy, including a generative pretrained transformer (GPT) in the sense of Zhao et al. 2023 as we recognize the growing interest in large language models (LLMs).

We expect that progressive difficulty MTL will enhance the performance of neural networks. By proposing and testing this MTL strategy, we hope to better understand the behavior of neural networks and establish links between their internal learning mechanisms and those of humans.

2 Background

In his review of previous work on MTL, Caruana (1997) introduced MTL as "an inductive transfer mechanism" that "improves generalization by leveraging the domain-specific information contained in the training signals of *related* tasks" (page 41). The motivation behind MTL is to divide and conquer: we break large problems into small ones (cf. Waibel et al., 1989). Subsequent work on MTL (a.o.: Kandemir et al., 2014; Jaques et al., 2017; Guo et al., 2020; Lu et al., 2020) also showed that similar tasks trained simultaneously can benefit from each other in terms of convergence time and overall accuracy. In particular, Lu et al. (2020) applied MTL to sentiment analysis and effectively improved the overall accuracy of variational autoencoders. In this paper, we use progressive difficulty subtasks for MTL, and we broaden the empirical ground of previous work beyond sentiment analysis to encompass other NLP tasks like text classification, unit segmentation, and syllogistic reasoning.

As for backbone models, among many others, Cerri et al. (2014) and Peng et al. (2018) applied neural networks to hierarchical text classification. The former offered a locally connected network approach, in which the prediction scores for the classi-

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop), pages 48–55

fication of the current label level are used as input to the classification of the next label level. The latter offered a convolutional neural network (CNN) approach, in which hierarchical dependencies among the labels are provided to a classifier with recursive regularization (Gopal and Yang, 2013). In this paper, we extend the experiments to additional types of neural networks, including the fully connected neural network (FCNN), the long short-term memory network (LSTM), and transformers.

Additionally, Conneau et al. (2017) employed deep CNNs for text classification, adopting ideas from VGG (Simonyan and Zisserman, 2015) and ResNet (He et al., 2016), using a small size of kernels for convolutional filters and adding residual connections to address degradation problems. With a similar motivation, Kim et al. (2017) employed deep LSTMs, which proved to be empirically remarkable on feature extraction. In this paper, as an extension of our study, we implement their deep neural networks and test how adding more layers to the model may affect the performance of progressive difficulty MTL. As another extension, we test how Vu et al.'s (2020) transfer learning (TL) may affect the strategy we propose.

Finally, in the field of LLMs, numerous recent studies (a.o.: Wei et al., 2022; Fan et al., 2023; Fei et al., 2023; Kim et al., 2023; Wang et al., 2023a,b,c) discussed a chain-of-though (CoT) strategy, which involves prompting the model to generate intermediate steps before arriving at a final answer. This strategy substantially enhanced model performance, and we believe that this strategy aligns perfectly with the strategy we propose in this paper: they "start simple." In this paper, we apply progressive CoT to syllogistic reasoning. Specifically, from an information flow perspective, we let the model be informed about the information of the main task using manually designed progressive difficulty subtasks.

3 Methods

We conduct two experiments. The first experiment works on sentiment analysis, text classification, and unit segmentation. The second experiment works on syllogistic reasoning.

3.1 Tasks and data sets

We conduct our experiments across a variety of tasks and data sets. A summary of the data sets is presented in Table 1.

Sentiment analysis. In a sentiment analysis task, a model is given a text and analyzes its discrete degree of positiveness/negativeness. For this task, we used a data set of coronavirus tweets (cf. Jelodar et al., 2020).[1] This data set contains 45k samples (41k training and 4k testing) with 3 L1 and 5 L2 labels. We split 4k samples from the training set for validation.

Text classification. In a text classification task, a model is given a text and classifies it into one of the predefined classes. For this task, we used data sets of Amazon product reviews and of DBPedia (Auer et al., 2007). The first data set contains 50k samples with 6 L1, 64 L2, and 510 L3 labels.[2] We concatenated the three levels of labels and dropped every sample that either belonged to a concatenated label having fewer than 64 samples or was shorter than 32 characters. We had around 40k samples left with 6 L1, 50 L2, and 147 L3 labels. The second data set contains 338k samples (241k training, 36k validation, and 61k testing) with 9 L1, 70 L2, and 219 L3 labels.[3]

Unit segmentation In a unit segmentation task, a model is given a text and segments it into predefined argumentative components. For this task, we used a data set of argumentative essays.[4] This data set contains 25k samples (15k training and 10k testing) with 15 labels. We split 3k samples from the training set for validation.

Syllogistic reasoning. In a syllogistic reasoning task, a model is given two or more statements and one or more conclusions and reasons about whether each conclusion logically follows from the statements. For this task, we used a data set of syllogism data.[5] We extracted the first conclusion in each sample and constructed a binary class of labels.

3.2 Experiment 1

Figure 1 sketches a demonstration of the fundamental structure of our proposed model in experiment 1 (cf. Lu et al., 2020) using three subtasks. In this model, the input batch is first passed into the

[1] https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification

[2] https://www.kaggle.com/datasets/kashnitsky/hierarchical-text-classification

[3] https://www.kaggle.com/datasets/danofer/dbpedia-classes

[4] https://www.kaggle.com/competitions/feedback-prize-2021

[5] https://www.kaggle.com/datasets/warcoder/syllogism-data

Data set	# of labels	# of samples	Sample length	Vocabulary size
Coronavirus tweets	3/5	45k	32	70k
Amazon product reviews	6/50/147	40k	96	42k
DBPedia	9/70/	338k	160	618k
Argumentative essays	3/7/15	25k	1024	30k
Syllogism data	2	65	N/A	40k

Table 1: Summary of the data sets.
Notes: The i-th number in the "# of labels" column represents the number of labels at the i-th level. At each level, fewer labels indicate lower difficulty. Syllogism data have no fixed sample length.

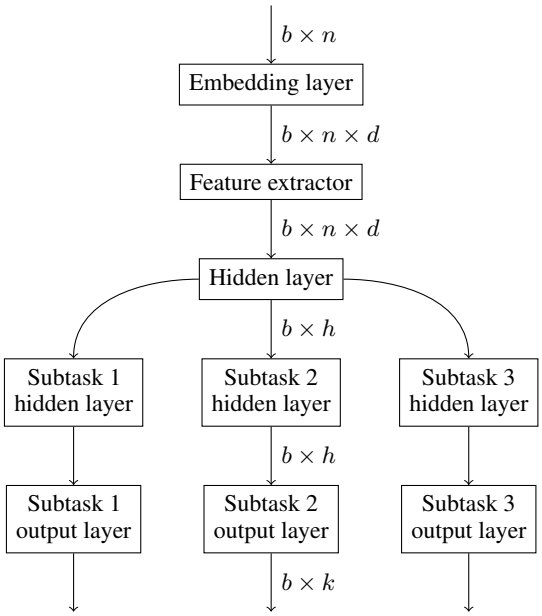

Figure 1: Model architecture in experiment 1.
Note: b is the batch size, n is the sample length, d is the embedding size, h is the hidden size, and k is the number of labels.

embedding layer, which then feeds the word embeddings to the feature extractor. What follows the feature extractor is a series of processing layers consisting of a shared hidden layer and, for each subtask, a subtask-specific hidden layer and an output layer. We implemented the hidden layers using the fully connected layer structure discussed by Zhang et al. (2015) and Conneau et al. (2017) and exploited 2048 units each. In order to accelerate convergence, we applied batch normalization (Ioffe and Szegedy, 2015) to every layer in the model except for the embedding layer and the output layers.

Embedding layer. We use word2vec to transform tokens into (32-dimensional) word embeddings (Mikolov et al., 2013a) in non-transformer-based models. Before training the backbone model, we pretrain these word2vec embeddings using skip-gram on each of the data set and freeze them afterward. The embedding layer does not get updated while pretraining the word embeddings or training the backbone model. According to Mikolov et al. (2013b) and Levy et al. (2015), a pretraining strategy like this initializes the model with semantically meaningful representations, which can capture the contextual and syntactic similarities between words and result in better generalization.

Feature extractor. For sentiment analysis and text classification, we employ and evaluate the FCNN (Popescu et al., 2009), the CNN (Kiranyaz et al., 2021), and the LSTM (Yu et al., 2019). For unit segmentation, considering its complexity, we use a transformer-based model. While BERT (Devlin et al., 2019) may appear to be the preferred option, its performance diminishes when applied to lengthy texts. To this end, we use Longformer (Beltagy et al., 2020), which not only performs better with long input sequences but also improves computational efficiency via dilated sliding windows for attention patterns.

Progressive difficulty subtasks. The definition of the difficulty of a task can be flexible and should be manually designed when tackling a specific task. In the scenarios of sentiment analysis, text classification, and unit segmentation, we define progressive difficulty subtasks as tasks that are otherwise identical but vary in their degree of coarseness. That is, the number of labels at different levels of difficulties follows this: $k_1 < k_2 < \cdots < k_t$, where k_s is the number of labels in the s-th subtask, and t is the number of subtasks. We assume that more labels mean higher difficulty, and the difficulty progressively increases from the subtask that has k_1 labels to the subtask that has k_t labels.

Loss function. The loss is a weighted sum of the loss of each subtask, and the loss function in every subtask is a softmax-loss function. Let $\hat{y}_s$ be the predicted output in the s-th subtask of a sample

with an actual output of y. Then, L_s, the loss in this subtask, is defined as in Equation 1.

$$L_s(\hat{y}_s, y) = -\sum_i (y_i \log \frac{e^{\hat{y}_{s,i}}}{\sum_j e^{\hat{y}_{s,j}}}) \quad (1)$$

Subsequently, the final loss L can be calculated using Equation 2, where w_s is the weight assigned to the s-th subtask.

$$L(\hat{y}, y) = \sum_s (w_s L_s(\hat{y}_s, y)) \quad (2)$$

3.3 Experiment 2

Recall that in a syllogistic reasoning task, the model needs to find out whether a given conclusion may be deduced from the given statements. In this task, we apply our methodology of progressive difficulty subtasks to the CoT strategy in LLMs, such as GPT. Specifically, instead of asking the LLM to go directly toward answering either true or false, we let it summarize the context and perform a basic, intuitive inference from the query in the meantime. In this scenario, summarizing and inferring are considered the progressive difficulty subtasks for syllogistic reasoning.

A demonstration of the prompts we used, encoded in Markdown, is presented in the red boxes below. The first red box contains the general system guidance, and the second one contains the dynamic prompt format of each query.

General prompt

```
# Background
Syllogisms are logical arguments of
statements using deductive
reasoning to arrive at a
conclusion.
# Task
You are a philosopher who conducts
syllogistic reasoning. You will be
given two or more statements
followed by a conclusion. Determine
whether the conclusion logically
follows from the given statements.
```

Specific prompt

```
# Query
## Statements
[...]
## Conclusion
[...]
```

The output format is shown in the blue boxes, including a baseline single prompt that fully relies on the zero-shot learning ability of LLMs in the first blue box and an improved version with the progressive CoT strategy in the second one.

Baseline output format

```
# Output
Please output the following:
## Result
True or False.
```

Progressive CoT output format

```
# Output
Please output the following:
## Summary
List all the relations between the
terms in the statements as in a
knowledge graph in the format of
(term 1, relation, term 2).
## Thought
Can the conclusion be inferred
from the statements following a
strict syllogistic logic?
## Result
True or False.
```

4 Results

In experiment 1, we used a batch size of 64 for coronavirus tweets and Amazon product reviews, 128 for DBPedia, and 4 for argumentative essays. We used a mini-batch SGD optimizer with a momentum of .9 and a fixed learning rate of .01. The models converged in around 15 epochs for coronavirus tweets and DBPedia, 30 epochs for Amazon product reviews, and 5 epochs for argumentative essays. In the meantime, baseline models were trained without MTL. Table 2 presents the overall accuracy in each task. We found that by training the model on both the main task and its simplified versions simultaneously, the performance of the model improved in all cases. The improvement is relatively less notable in unit segmentation, but the difference is statistically significant with a p value less than .001.

As an extension to experiment 1, we tried to stack more layers inside the feature extractor used in the sentiment analysis and text classification tasks and see if they can boost the performance of MTL with progressive difficulty subtasks. As

Task	Data set	Feature extractor	Baseline	Progressive difficulty
Sentiment analysis	Coronavirus tweets	FCNN	35.91	**38.70**
		CNN	37.78	**41.91**
		LSTM	46.36	**53.55**
Text classification	Amazon product reviews	FCNN	17.03	**19.16**
		CNN	25.72	**26.77**
		LSTM	32.60	**41.53**
	DBPedia	FCNN	83.17	**85.11**
		CNN	90.57	**90.68**
		LSTM	91.28	**92.26**
Unit segmentation	Argumentative essays	Longformer	70.51	**70.58**

Table 2: Mean overall accuracy over three repetitions of experiment 1 (%).

mentioned in the background section, for CNN, We implemented the deep neural network discussed in Conneau et al. 2017, which contained 17 convolutional layers followed by max pooling and fully connected layers. Additionally, we implemented the deep LSTM discussed in Kim et al. 2017 with 8 LSTM layers, which added residual connections between every two layers in the middle six layers. Table 3 presents the results. We observed that while trained using progressive difficulty MTL, the CNN may be benefited from stacking more layers, whereas the LSTM does not improve as much.

Further, following Vu et al. (2020), we tested the effect of transfer learning to progressive difficulty MTL. We added name entity recognition and text classification to the task pipeline of unit segmentation and applied progressive difficulty MTL to both of them. For name entity recognition, we used the CoNLL-2003 data set (Tjong Kim Sang and de Meulder, 2003), and for text classification, we used the aforementioned DBPedia data set. We present the results in Table 4, where we do not see a positive effect of transfer learning on MTL with progressive difficulty subtasks.

Last but not least, we conducted experiment 2 using GPT 3.5. The results are in Table 5, where we can see that progressive CoT achieved notable improvement, without an extensive prompt design.

5 Discussion

As can be seen in Table 2 and Table 5, progressive difficulty MTL improved the model performance in all four NLP tasks. This result may suggest that neural networks can extract common features and internalize shared representations from progressive difficulty subtasks. It may also suggest that they can adapt to increasingly complex problems if they are trained in a structured manner. These capabilities are akin to human learning, where we apply our knowledge from simpler related problems to more complex problems.

Results in Table 3 show that deep neural networks can improve the performance of our proposed model when the feature extractor is a CNN but not when it is an LSTM. We attribute this difference to the distinction in their field of view, since CNNs are structured so that each layer captures increasingly complex features, whereas LSTMs have an architectural bottleneck. The ability to capture increasingly complex features is especially crucial within the context of progressive difficulty MTL, as learning from features of different levels of complexity can effectively benefit a neural network's performance. This inference leads us to conclude that our strategy prefers a neural network that has a large field of view.

Table 4 indicates that TL with complementary data sets cannot improve the performance of MTL with progressive difficulty subtasks. This result suggests that progressive difficulty MTL suits better for configurations where the goal of all subtasks is concentrated and uniform. This observation is on par with previous findings about MTL, which is believed to perform better when trained using more related subtasks (cf. Caruana, 1997).

6 Conclusion

Inspired by how humans learn, we proposed an MTL strategy using progressive difficulty subtasks and discovered that this strategy improved the performance of various neural networks on various NLP tasks. We also found out that our strategy worked better with neural networks having a larger field of view and with subtasks sharing a common, focused goal. We stipulate that the internal learning mechanisms of neural networks are akin to human learning in the sense that it can apply its knowledge from simpler tasks to more complex tasks.

Task	Data set	Feature extractor	Label level	Shalow	Deep
Sentiment analysis	Coronavirus tweets	CNN	L1	61.29	**63.21**
			L2	38.52	**41.91**
		LSTM	L1	**70.35**	59.89
			L2	**53.55**	39.57
Text classification	Amazon product reviews	CNN	L1	70.52	**75.53**
			L2	43.04	**49.46**
			L3	26.77	**34.13**
		LSTM	L1	75.79	**77.96**
			L2	51.95	**53.56**
			L3	**41.53**	41.12
	DBPedia	CNN	L1	**97.52**	97.39
			L2	93.86	**94.28**
			L3	90.68	**91.07**
		LSTM	L1	**97.67**	96.72
			L2	**94.92**	93.71
			L3	**92.26**	90.39

Table 3: Mean overall accuracy over three repetitions of sentiment analysis and text classification using progressive difficulty MTL and shallow vs. deep neural networks (%).

Label level	No TL	CoNLL-2003	DBPedia
L1	**78.18**	78.17	78.16
L2	**71.51**	**71.51**	71.44
L3	**70.58**	70.57	70.53

Table 4: Mean overall accuracy over three repetitions of unit segmentation using progressive difficulty MTL and transfer learning (%).

Baseline	Progressive CoT
72.99 (.813)	**78.16** (.813)

Table 5: Mean overall accuracy (standard deviation) over three repetitions of experiment 2 (%).

Limitations

In experiment 1, we opted for a model without MTL, but it could be argued that for a fair comparison, the baseline model should also incorporate MTL. We are open to suggestions regarding what kind of subtasks should be included in the alternative baseline models.

In experiment 2, we could not confirm whether GPT 3.5 was multitasking in parallel as in the other three NLP tasks rather than in sequence. This is due to the nature of GPT, especially its large size, which makes it challenging to deploy with the limited computing resources available to us. We welcome feedback on ways to clarify this matter.

Another limitation in our work is that we did not explore the extent to which information is shared among progressive difficulty subtasks. Much of our effort focused on demonstrating the applicability and practicality of our proposed strategy, and we leave the scope of information sharing as a future research question.

Acknowledgments

We owe our deepest gratitude to Mohit Iyyer and Brendan O'Connor for their constructive feedback. We also thank our anonymous NAACL SRW reviewer for helpful comments and insightful remarks. All errors are our own.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. In *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 2007, Proceedings*, pages 722–735, Busan, Korea. Springer.

Iz Beltagy, Matthew Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Computing Research Repository*, arXiv:2004.05150. Version 2.

Rich Caruana. 1997. Multitask Learning. *Machine Learning*, 28(1):47–75.

Ricardo Cerri, Rodrigo Barros, and André de Carvalho. 2014. Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, 80(1):39–56.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann LeCun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1:*

Long Papers), pages 1107–1116, Valencia, Spain. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long and Short Papers)*, pages 4171–4186, Minneapolis, MN, USA. Association for Computational Linguistics.

Caoyun Fan, Jidong Tian, Yitian Li, Wenqing Chen, Hao He, and Yaohui Jin. 2023. Chain-of-thought tuning: Masked language models can also think step by step in natural language understanding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14774–14785, Singapore. Association for Computational Linguistics.

Hao Fei, Bobo Li, Qian Liu, Lidong Bing, Fei Li, and Tat-Seng Chua. 2023. Reasoning implicit sentiment with chain-of-thought prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1171–1182, Toronto, Canada. Association for Computational Linguistics.

Siddharth Gopal and Yiming Yang. 2013. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 257–265, Chicago, IL, USA. Association for Computing Machinery.

Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. 2020. Learning to branch for multi-task learning. *Proceedings of Machine Learning Research*, 119:3854–3863.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, Las Vegas, NV, USA. Institute of Electrical and Electronics Engineers.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of Machine Learning Research*, 37:448–456.

Natasha Jaques, Ognjen Rudovic, Sara Taylor, Akane Sano, and Rosalind Picard. 2017. Predicting tomorrow's mood, health, and stress level using personalized multitask learning and domain adaptation. *Proceedings of Machine Learning Research*, 66:17–33.

Hamed Jelodar, Yongli Wang, Rita Orji, and Shucheng Huang. 2020. Deep sentiment classification and topic discovery on novel coronavirus or COVID-19 online discussions: NLP using LSTM recurrent neural network approach. *IEEE Journal of Biomedical and Health Informatics*, 24(10):2733–2742.

Melih Kandemir, Akos Vetek, Mehmet Gönen, Arto Klami, and Samuel Kaski. 2014. Multi-task and multi-view learning of user state. *Neurocomputing*, 139:97–106.

Jaeyoung Kim, Mostafa el Khamy, and Jungwon Lee. 2017. Residual LSTM: Design of a deep recurrent architecture for distant speech recognition. In *Proceedings of Interspeech 2017*, pages 1591–1595, Stockholm, Sweden. International Speech Communication Association.

Seungone Kim, Se Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 2023. The CoT collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12685–12708, Singapore. Association for Computational Linguistics.

Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel Inman. 2021. 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 151:107398.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Zachary Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57.

Marsha Lovett, Michael Bridges, Michele DiPietro, Susan Ambrose, and Marie Norman. 2023. *How Learning Works: Eight Research-Based Principles for Smart Teaching*, 2nd edition. Jossey-Bass, Hoboken, NJ, USA.

Guangquan Lu, Xishun Zhao, Jian Yin, Weiwei Yang, and Bo Li. 2020. Multi-task learning using variational auto-encoder for sentiment classification. *Pattern Recognition Letters*, 132:115–122.

Tomáš Mikolov, Kai Chen, Gregory Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *Computing Research Repository*, arXiv:1301.3781. Version 3.

Tomáš Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, volume 2, pages 3111–3119, Lake Tahoe, NV, USA. Curran.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-CNN. In *Proceedings of the 2018 World Wide Web Conference*, pages 1063–1072, Lyon, France. International World Wide Web Conferences Steering Committee.

Marius-Constantin Popescu, Valentina Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. 2009. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588.

Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. 2022. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16:1–85.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. *Computing Research Repository*, arXiv.1409.1556. Version 6.

Erik Tjong Kim Sang and Fien de Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, Edmonton, AB, Canada. Association for Computational Linguistics.

Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. 2020. Exploring and predicting transferability across NLP tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7882–7926, online. Association for Computational Linguistics.

Alexander Waibel, Hidefumi Sawai, and Kiyohiro Shikano. 1989. Modularity and scaling in large phonemic neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):1888–1898.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023a. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2717–2739, Toronto, Canada. Association for Computational Linguistics.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023b. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Yiming Wang, Zhuosheng Zhang, and Rui Wang. 2023c. Element-aware summarization with large language models: Expert-aligned evaluation and chain-of-thought method. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8640–8665, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 24824–24837, New Orleans, LA, USA and online. Curran.

August Martin Wildberger. 1994. Alleviating the opacity of neural networks. In *Proceedings of 1994 IEEE International Conference on Neural Networks*, volume 4, pages 2373–2376, Orlando, FL, USA. Institute of Electrical and Electronics Engineers.

Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. 2019. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Computation*, 31(7):1235–1270.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, volume 1, pages 649–657, Montreal, QC, Canada. MIT Press.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *Computing Research Repository*, arXiv:2303.18223. Version 13.