Automated Fact Checking System For Climate Science Claims

Mon1PM_Group5

Abstract

The proliferation of misinformation in climate science necessitates robust automated fact-checking systems to ensure the accuracy of public discourse. This project aims to develop a system capable of classifying climate science claims, based on evidence retrieved from a large corpus. We approached evidence retrieval as a classification problem, utilizing a Siamese network to compute cosine similarity between claim-evidence pairs, and trained it with contrastive loss and cyclic learning rates. For prediction, a top-k approach was employed. This model achieved a dev F1-score of 0.665. For claim classification, transformer model was trained with focal loss to address class imbalance. The model was evaluated using a traditional ML approach, resulting in a dev accuracy of 0.487. The results significantly surpassed the baseline scores, validating our system's capability to enhance the reliability of fact-checking, contributing to the mitigation of false rumours.

1 Introduction

The increasing prevalence of online false information, such as fake news and rumors, has exposed people to various misleading narratives not only about their daily lives but also about critical issues like climate science (Elhadad et al., 2020). Climate change denial and pseudo-scientific narratives can undermine environmental policy and public awareness. Such misleading information can be used as political and ideological warfare to manipulate public perceptions (Ibrishimova and Li, 2020). Hence, it is imperative to develop an automated fact-checking system to guarantee reliable and validated information.

This is challenging due to the need to accurately retrieve and evaluate relevant evidence from vast corpuses, and requires sophisticated natural language understanding and reasoning. Traditional fact-checking approaches, such as rule-based systems and keyword matching, are insufficient for the complexity of climate science claims. Although recent progress in ML and NLP has shown promise (Hanselowski et al., 2019), challenges like class imbalance and evidence relevance still persist. Most approaches also suffer from large memory needs and limited flexibility (Chakrabarty et al., 2018).

We address these challenges through a two-stage process: evidence retrieval and claim classification. For the former, we used a Siamese network to compute cosine similarity between claim-evidence pairs, trained with contrastive loss and cyclic learning rates. We used a top-k method for prediction, achieving a dev F1-score of 0.665. For claim classification, we employed a transformer model trained with focal loss to address class imbalance. This model achieved a dev accuracy of 0.487. Our final harmonic mean of 0.546 demonstrates significant improvement over the baseline score of 0.344, validating our approach's effectiveness.

2 Preprocessing Steps

First, we preprocessed the raw texts using multiple steps. Text is initially converted to lowercase for uniformity. Contractions are replaced to ensure consistency, followed by tokenization for feature extraction and vocabulary creation. Punctuation is removed to simplify text and emphasize key content. Tokens are stemmed to reduce vocabulary size and improve training efficiency, and then converted into integer sequences and padded to a standardized length of 20 (average English sentence length (Gina, 2023)), ensuring uniform input for the models. Notably, stop words are retained as they are crucial for distinguishing between class labels.

3 Main Approach

3.1 Baseline Model

For the baseline to compare our final outputs against, we used the 'dev-claims-baseline.json' file provided in the project repository (Han, 2024).

3.2 Evidence Retrieval

In this task, for a given claim c_i, we select the evidences e_i from a given corpus $e = \{e_1, \ldots, e_j\}$ whose predicted relevance probability $p(c, e_i)$ is among the top-k probabilities.

3.2.1 Siamese Network

For the evidence retrieval task, we use a Siamese network (Neculoiu et al., 2016), a type of neural network architecture that involves using two identical branches to process two different input samples. These branches share the same weights and are used to compare or combine outputs for specific tasks like similarity calculation.

We are using this model for two key reasons (Neculoiu et al., 2016). Firstly, they are great at assessing the semantic relationships between text inputs, which is crucial for evaluating evidence's relevance to a claim. Secondly, by using pairwise comparison, they can efficiently determine the similarity or relevance of text passages without needing exhaustive search methods or manual scaling.

In our model (Figure 1), we first pass a claim text and an evidence text through embedding and Bi-LSTM layers to process the sequences and encode them. Bi-LSTM allows to capture contextual dependencies from both directions and can produce a more meaningful output (Zvornicanin, 2024). Then, we compute their cosine similarity score (Han et al., 2012), which is a normalised score between [-1, 1] so there is no issue of varied scale. For novelty, we added residual connections between the encoded texts and similarity score so the model can leverage both the intrinsic features of the texts and the derived similarity measure for accurate predictions. The merged output is passed through hidden layers (with dropout) and batch normalisation (for stability), before outputting the predicted probability for the relevance of the texts.

3.2.2 Contrastive Loss Function

To train our model, we applied the contrastive loss function (Hadsell et al., 2006), commonly used in Siamese networks and related architectures for tasks such as text similarity, image similarity, and more. It inspires the model to learn embeddings in a way that similar inputs are mapped closer together in the embedding space, while dissimilar inputs are pushed apart by at least a fixed margin.

In mathematical terms, this can be written as: $L(y, \hat{y}) = \frac{1}{2}(y_i \cdot d_i^2 + (1 - y_i) \cdot (\max(m - d_i, 0))^2)$ where:

- y_i = true label for the pair of inputs, indicating similarity (1) or dissimilarity (0).
- $\hat{y}_i$ = predicted output of the model.
- d_i = distance between input embeddings.
- m = minimum desired separation between embeddings of dissimilar pairs.

The reason we used this custom loss lies in its unique capabilities (Hadsell et al., 2006) suited for our task. First, by penalizing the model when similar inputs are not close enough and vice versa, it encourages the model to learn discriminative embeddings and capture the semantic similarity, helping to determine if a claim-evidence pair is relevant. Secondly, the margin allows for fine-tuning the trade-off between the separation of dissimilar pairs and the compactness of similar pairs in the embedding space. For our task, we used a margin of 1, as the outputs are probabilities in [0,1] range. Finally, the hinge-like loss function makes it robust to class imbalance, penalising the model for errors in predicting the dissimilarity of pairs, which is particularly useful as the irrelevant claim-evidence pairs vastly outnumber the similar pairs.

3.2.3 Cyclical Learning Rate (Triangle2)

When training neural networks, traditional fixed learning rates (LR) may lead to slow convergence or sub-optimal solutions (Brownlee, 2019), especially when dealing with complex tasks and extensive datasets, such as ours with more than 1.2 million evidences. Thus, we used a cyclical LR, namely the "triangle2" policy (Figure 2), that involves cyclically increasing and decreasing the LR within a specified range (Smith, 2017).

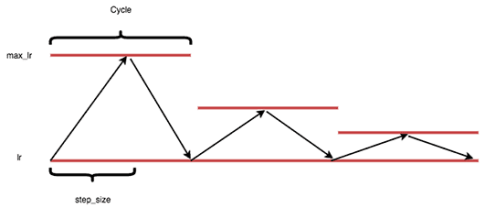

Figure 2: Cyclical Learning Rate Scheduler (Triangle2)

This dynamic LR scheduler provides several key benefits for our complex model (Smith, 2017). First, by cycling, the model can escape from sharp minima and explore a broader range of solutions, potentially leading to faster convergence. It also enhances robustness of the training process by allowing the model adapt to different regions of the loss landscape, making it less sensitive to the initial choice of LR, and reducing the need for extensive

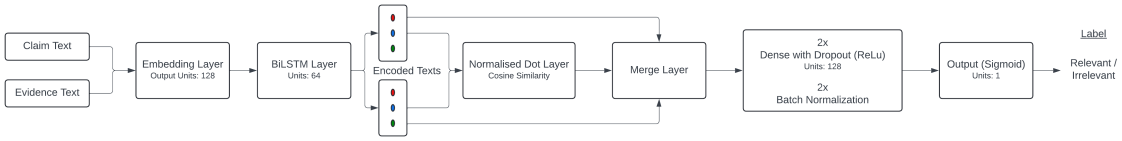

Figure 1: Architecture of the Developed Siamese Model for Evidence Retrieval

hyperparameter search. Finally, the cyclic nature can act as a form of regularisation, preventing our model from overfitting to local minima.

To tune the scheduler, we used the same parameters as (Smith, 2017). We defined the LR bounds between [0.0001, 0.01]. The rate linearly oscillates between these bounds over a cycle size which we specified as 2 epochs. The use of "triangle2" policy allows to cut the max bound by half in every cycle to let the LR break out of saddle points and at the same time decrease it, enabling us to descend into lower loss areas of the loss landscape.

3.2.4 Training and Prediction

For training the model, we formalised the problem as a classification task with 1 being relevant and vice versa. In an ideal system, we would pair each claim with each evidence, but given the large number of evidences, we pair a fixed number of evidences per claim, n, (including all relevant), such that total evidence per claim = $\min(\max(2 \cdot \text{relevant evidences}, n), \text{available evidences})$ to ensure a surplus of irrelevant evidence in most cases like in a real world. We also used early stopping to prevent overfitting during training.

For making predictions, we used a top-k method based on predicted probabilities. If the predicted probability of an evidence being relevant for a particular claim is among the k highest predicted probabilities, we accept that evidence as being relevant. This is to ensure uniform input data shape for the classification model in the next step.

3.3 Claim Classification

In this task, for a given claim c_i and retrieved relevant evidences $e_i = \{e_1, \ldots, e_k\}$, we classify the claim into one of {SUPPORTS, REFUTES, NOT_ENOUGH_INFO, DISPUTES}.

3.3.1 Transformer-Encoder Model

LSTM and GRU are effective for NLP tasks by addressing vanishing gradients, but their recurrent structure challenges parallelization and handling long clauses (Zeyer et al., 2019). To overcome these, we used a transformer model (Vaswani et al., 2017), which efficiently manages long-distance dependencies and inherently learns meaningful representations directly from data, without relying on external embeddings. Additionally, we benefit from the model's self-attention mechanism, allowing for efficient parallel processing and faster training.

A Transformer-Encoder, part of the full transformer architecture, is sufficient for our task focused on classification rather than sequence generation (Yüksel et al., 2019). By removing the decoding component, the model is simpler and more computationally efficient, making it easier to implement, interpret, and debug. This streamlined approach provides an effective solution for claim classification without unnecessary complexity.

3.3.2 Sparse Categorical Focal Loss

The training data exhibits class imbalance, with 'SUPPORTS' and 'NOT_ENOUGH_INFO' overrepresented, and the others underrepresented. To address this without losing information through data level balancing techniques given the small training dataset size, we instead used a classifier level technique, focal loss (Lin et al., 2020).

Focal loss is effective for imbalanced datasets and also performs well with noisy data and hard-to-classify samples (Shi et al., 2018). It assigns higher weights to misclassified examples, particularly those that are more challenging, thus helping the model to better recognize less common classes.

Mathematically, it can be expressed as: $L(p_i) = (\alpha(1-p_i)^\gamma \log(p_i))$ where:

- p_i = predicted probability
- γ = focusing parameter
- α = weight factor for the classes

For γ which adjusts the rate of down-weighting easy samples, we used a value of 2, same as (Shi et al., 2018), and set $\alpha = 0.25$, reducing the impact of misclassifications for majority class by 25%.

3.3.3 Training Process

The input tokens are transformed into embedding vectors to match the model's format, which uses positional embeddings to encode the sequence order. The model consists of multiple identical layers,

each with a multi-head self-attention layer and a feed-forward network, interconnected with residual connections and normalization, followed by a global average pooling and softmax output. We set 'return_best_weights=True' with early stopping to retrieve the weights of the model with the best validation accuracy before overfitting had any effect.

4 Experimentation

For evidence retrieval, our main parameters of interest were the choice of n and k to see how they influenced training metrics, and so were fiddled with to enhance model performance while keeping other typical hyperparameters constant throughout.

Choosing n had a trade-off between training time and model performance (F-score). Although a high n would allow to sample a large number of evidence instances to train the model, improving its generalizability, it will take more time to train it. While defaulting $k = 5$, we tried different n to test the trade-off and inform our decision (Table 1).

n	Train Time/Epoch (s)	Dev F1 score
20	21	0.475
40	43	0.665
50	185	0.675
80	477	0.691
100	768	0.711

Table 1: Model Performance for Different n

For us, $n = 40$ gave the best balance of the metrics compared to its higher counterparts, which had minimal performance improvements.

Next, the trade-off between precision and recall hinges on the choice of k in the top-k method. A larger k increases the number of retrieved items, potentially enhancing recall but likely at the expense of precision due to the inclusion of more irrelevant items and vice versa (Manning et al., 2008). Setting $n = 40$, we trained with different k values (Table 2), and selected $k = 5$ with the best F-score.

k	Precision	Recall	F1 score
2	0.498	0.614	0.550
3	0.571	0.765	0.654
4	0.569	0.791	0.662
5	0.551	0.866	0.673
6	0.488	0.889	0.630
7	0.475	0.985	0.641

Table 2: Model Training Performance for Different k

For the claim classification task, to enhance the model's ability to capture local patterns, we initially tried adding a 2D convolutional layer, which is effective at processing local information by using a sliding window (Gulati et al., 2020). However, despite a sharp increase in training accuracy, validation accuracy remained below 0.4, indicating that the model did not learn effectively and overfit. This is likely due to excessive zero padding of inputs during preprocesing which can reduce training efficiency by forcing the model to learn unnecessary features from the padded parts (Zhang et al., 2023).

For hyperparameter tuning, we aimed to optimize the number of Transformer-Encoder layers (nl), embedding dimension (ed), number of attention heads (nh), feed-forward dimension (ffd), dropout rate (dr), and learning rate (lr). Despite using random search with Keras Tuner, finding locally optimal settings required significant computation time due to model complexity. Thus, we manually tried various parameter combinations (Table 3).

Hyperparameter	Val Acc	Dev F1 score
nl = 4, ed = 64, nh = 4, ffd = 128, dr = 0.1, lr = 1e-3	0.41	0.26
nl = 5, ed = 64, nh = 6, ffd = 256, dr = 0.3, lr = 1e-4	0.44	0.15
nl = 4, ed = 128, nh = 8, ffd = 256, dr = 0.2, lr = 1e-4	0.44	0.32
nl = 10, ed = 128, nh = 4, ffd = 640, dr = 0.5, lr = 1e-4	0.46	0.35
nl = 3, ed = 128, nh = 8, ffd = 128, dr = 0.2, lr = 1e-4	0.49	0.40

Table 3: Hyperparameter Tuning Results

Same as before, balancing the precision-recall trade-off was challenging. Hence, we tried to ensure high average dev F-score and validation accuracy, and ultimately selected the last parameter combination from Table 3 with best performance.

5 Results

Table 4 presents the evidence retrieval results, with both datasets achieving a little over 0.5 precision and 0.85 recall. This indicates that only half of the retrieved evidence is relevant, which was ex-

pected given the use of top-k, forcing the retrieval of 5 pieces of evidence per claim, including some irrelevant ones. Despite this, the high recall demonstrates the model's strong capability in identifying most relevant evidence, crucial for the claim classification task that requires high recall to ensure comprehensive coverage of evidence for better predictions (Samarinas et al., 2021).

	Precision	**Recall**	**F1 score**
train set	0.551	0.866	0.673
dev set	0.534	0.883	0.665

Table 4: Evidence Retrieval Performance

Moving on to the classification results, the transformer model only predicts 112 claims as "SUPPORTS and 42 as "NOT_ENOUGH_INFO" for the dev set, resulting in a biased output (Figure 3). Table 5 shows a dev accuracy of 0.487, meaning we correctly predict about 49% of the true labels.

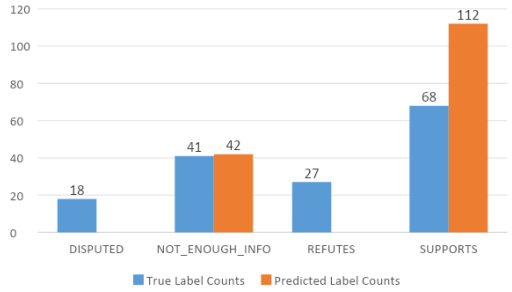

Figure 3: True and Predicted Label Counts for Dev Set

	P	**R**	**F1**
DISPUTED	0.00	0.00	0.00
NOT_ENOUGH_INFO	0.43	0.44	0.43
REFUTES	0.00	0.00	0.00
SUPPORTS	0.51	0.84	0.63
Accuracy		0.487	
Macro avg	0.23	0.32	0.27

Table 5: Dev Set Classification Report

The poor accuracy was expected due to the inclusion of irrelevant evidence in the previous task, which the transformer model treated as relevant, causing confusion. The bias likely stems from inherent biases in the dataset itself which even focal loss couldn't rectify. Previous studies indicate that dataset biases can influence classifiers, resulting in biased outcomes (Chakraborty et al., 2021). Another reason is our use of early stopping. Figure 4 shows validation loss increasing and training loss decreasing around epoch 7, indicating overfitting. Early stopping prevents the model from learning random regularities, albeit at the cost of improving its problem-solving ability (Ying, 2019). We used best weights of the model before overfitting sets in.

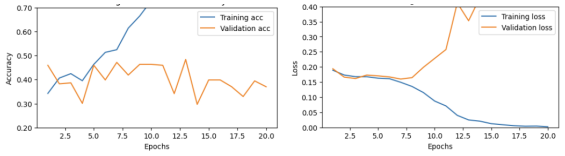

Figure 4: Training and Validation Metrics over Epochs

Training the model for 20 epochs increased the recall for "DISPUTED" and "REFUTES" to 0.17 and 0.15 respectively (not significant), but also brought accuracy down to 0.36, indicating overfitting, making the outcome less reliable than before.

Nonetheless, with weighted evidence retrieval F-score of 0.622 and classification accuracy of 0.487, we attain a moderate harmonic mean of 0.546. This is substantially higher than the baseline of 0.344. However, there is still room for improvement as the result is nowhere near the top scores of the original FEVER challenge (CodaLab, 2018).

6 Conclusion

This project investigated the efficacy of automated fact-checking systems with more focus on evidence retrieval as high quality evidence is likely to result in improved performance of the claim verification model. We employed a Siamese network for evidence retrieval and a Transformer-Encoder model for label classification, complemented by various supportive techniques to enhance performance.

However, limitations in the experiment impacted our findings. We were restricted from utilizing pre-trained techniques such as Word2Vec or BERT thereby increasing the challenge for our model in word distinction and contextual representation. Incapability of using rule-based techniques refrained us from manually filtering the evidence corpus to contain only climate-related texts, leading to a large dataset. Additionally, limited GPU resources extended our training duration. Future work could entail use of advanced filtering and data-balancing mechanisms, and techniques like entity-recognition for better context comprehension, given the availability of better computational resources.

Despite these challenges, our moderate final harmonic mean score suggests the system's overall effectiveness in fulfilling the project objectives.

7 Team Contributions

1. **Rafsan Al Mamun:**
   - **General:** Acted as the project manager, overseeing the whole project tasks throughout, delegating tasks to all members of the team, and troubleshooting any technical issues members faced. Acted as the team liaison facilitating communication both within the group and with tutors on their behalf. Helped with planning the project tasks and high-level models to be used for the main approach.
   - **Coding:** Performed data preprocessing to get the data ready for modelling. Developed and trained the evidence retrieval model from scratch, including the contrastive loss function and cyclic LR, and conducted the necessary experimentation. Built the basic transformer model for others to get started.
   - **Report:** Wrote the abstract, baseline model and evidence retrieval model (along with its experiments) sections. Edited and merged all the sections written by others into the final report.
   - **Presentation:** Created the slide template for the team to edit. Designed the slides for the evidence retrieval model. Edited the slides made by others to ensure coherent design.
2. **Mengyi Dai:**
   - **Coding:** Created several experiments to choose settings for transformer model.
   - **Report and Presentation:** Analyzed the results of the models with scrutiny. Investigated the reasons behind the outcomes with substantial background research. Discussed project limitations and avenues for future work.
3. **Tao Peng:**
   - **Coding:** Developed a Convolutional LSTM model for classification task, but was scrapped due to poor performance.
   - **Report and Presentation:** Introduced the topic, including its importance and motivation behind it. Explained the primary objectives, research question and the data preprocessing steps.
4. **Yuke Wang:**
   - **Coding:** Constructed the Transformer-Encoder model and conducted multiple experiments with various refinement techniques. These included exploring suitable loss functions, testing convolutional layers, experimenting with hyperband, random search, and manual tuning of hyperparameters, and investigating L1 and L2 regularization.
   - **Report:** Provided detailed explanation of the transformer model and experimental details, along with critical evaluation of the experiments.
   - **Presentation:** Explained the classification task, justified use of Transformer-Encoder and focal loss, and outlined the training process.

References

Jason Brownlee. 2019. How to configure the learning rate when training deep learning neural networks. Machine Learning Mastery.

Tuhin Chakrabarty, Tariq Alhindi, and Smaranda Muresan. 2018. Robust Document Retrieval and Individual Evidence Modeling for Fact Extraction and Verification. In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 127–131, Brussels, Belgium. Association for Computational Linguistics.

Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. 2021. Bias in machine learning software: why? how? what to do? In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE '21. ACM.

CodaLab. 2018. Fact extraction and verification (fever) challenge.

Mohamed K. Elhadad, Kin Fun Li, and Fayez Gebali. 2020. Detecting misleading information on COVID-19. *IEEE Access*, 8:165201–165215.

Gina. 2023. How many words should be in a sentence? Insights by Language Tool.

Anmol Gulati, Chung-Cheng Chiu, James Qin, Jiahui Yu, Niki Parmar, Ruoming Pang, Shibo Wang, Wei Han, Yonghui Wu, Yu Zhang, and Zhengdong Zhang, editors. 2020. *Conformer: Convolution-augmented Transformer for Speech Recognition*. arXiv.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

Caren Han. 2024. Comp90042_2024. GitHub repository.

Jiawei Han, Micheline Kamber, and Jian Pei. 2012. 2 - Getting to know your data. In *Data Mining*, third edition, The Morgan Kaufmann Series in Data Management Systems, pages 39–82. Morgan Kaufmann, Boston.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2019. Ukp-athene: Multi-sentence textual entailment for claim verification.

Marina Danchovsky Ibrishimova and Kin Fun Li. 2020. A machine learning approach to fake news detection using knowledge verification and natural language processing. In *Advances in Intelligent Networking and Collaborative Systems*, pages 223–234, Cham. Springer International Publishing.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2020. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, page 148–157. 2016 Association for Computational Linguistics.

Chris Samarinas, Wynne Hsu, and Mong Li Lee. 2021. Improving evidence retrieval for automated explainable fact-checking. *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, page 84–91.

Yunsheng Shi, Jun Meng, Jian Wang, Hongfei Lin, and Yumeng Li. 2018. A normalized encoder-decoder model for abstractive summarization using focal loss. In *Natural Language Processing and Chinese Computing*, pages 383–392, Cham. Springer International Publishing.

Leslie N. Smith. 2017. Cyclical learning rates for training neural networks. *arXiv*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Xue Ying. 2019. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, 1168(2):022022.

Atıf Emre Yüksel, Yaşar Alim Türkmen, Arzucan Özgür, and Berna Altınel. 2019. Turkish tweet classification with transformer encoder. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1380–1387, Varna, Bulgaria. INCOMA Ltd.

Albert Zeyer, Parnia Bahar, Kazuki Irie, Ralf Schlüter, and Hermann Ney. 2019. A comparison of transformer and LSTM encoder decoder models for ASR. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Zhiyi Zhang, Pengfei Zhang, Zhuopin Xu, and Qi Wang. 2023. Reduce computational complexity for convolutional layers by skipping zeros. In *2023 IEEE 30th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, pages 347–356.

Enes Zvornicanin. 2024. Differences between bidirectional and unidirectional LSTM. Baeldung on Computer Science.