Convolution Revolution: Wiener Filters for Embedded Text Comparison

Andrei-Cristian Dănilă

Imperial College London
Department of Earth Science and Engineering

Supervisors:
Dr. Lluis Guasch
Prof. Gerard Gorman
Deborah Pelacani Cruz

IMPERIAL

Abstract

Traditional approaches for comparing embedded sequences of text in Natural Language Processing (NLP) have largely remained unchanged since the introduction of the Transformer by [VSP+23]. This paper proposes a novel integration of Wiener filters within Transformer models to enhance sequence comparison. By leveraging the noise-reduction and global correlation properties of Wiener filters, I redefine the attention mechanism and loss function, aiming to increase the precision of contextual embeddings in NLP tasks. This novel approach bridges signal processing techniques with deep learning, potentially advancing Transformer performance in text-based applications through a more holistic embedding comparison strategy. The results show that there is potential for these techniques to break state-of-the-art benchmarks.

1 Introduction and Related Work

Recently, the field of natural language processing (NLP) has witnessed groundbreaking advancements, particularly since the Transformer model introduced in [VSP+23]. The Transformer's innovative self-attention mechanism has redefined the landscape of NLP by enabling efficient and scalable sequence processing, surpassing the capabilities of recurrent and convolutional neural networks. Concurrently, novel approaches in data comparison techniques, such as those proposed by Cruz et al. in 2023 ([CSB+23]), have emphasized the importance of maintaining global spatial correlations in data analysis, challenging the adequacy of traditional metrics like mean squared error (MSE). This paper seeks to integrate these advancements by exploring the potential benefits of convolutional similarity metrics, specifically Wiener filters, within the attention mechanism and the loss function of the Transformer model. This integration aims to enhance the model's ability to capture spatial and contextual correlations between embeddings, ultimately improving the awareness of data comparisons in NLP tasks and enhancing model performance.

1.1 Wiener Filters for Data Comparison

Recent advancements in data comparison techniques have highlighted the limitations of traditional methods like mean squared error (MSE), particularly in capturing global spatial structures in data.

Wiener filters, originally introduced by Norbert Wiener [Wie49], are widely used in image processing for denoising and deblurring tasks. These filters operate by minimizing the mean squared error (MSE) between the original signal (or image) and the estimated signal. In essence, a Wiener filter acts as an optimal linear filter that balances noise reduction with the preservation of important signal features. By modeling both the signal and the noise characteristics, the Wiener filter effectively suppresses unwanted noise while retaining the crucial details of the original image, leading to a clearer and more accurate reconstruction.

In "Convolve and Conquer: Data Comparison with Wiener Filters" ([CSB+23]) by Cruz et al. (2023), the authors propose a novel approach using Wiener filters to measure data similarity. The convolutional nature of Wiener filters enables a more holistic comparison of data samples by maintaining global correlations, thereby addressing the inadequacies of MSE which assumes data points are independent and of equal importance.

By applying the Wiener filter in a convolutional manner, Cruz et al. (2023) demonstrated its utility in maintaining global correlations within data samples, addressing the shortcomings of traditional point-wise similarity measures. This method has been shown to improve resolution and perceptual quality in reconstructed images, suggesting broader applicability in various machine learning tasks where data integrity and holistic comparison are critical. The research underscores the potential of Wiener filters to serve as a versatile tool for data comparison across various machine learning tasks, promoting a shift from local to global data analysis perspectives.

1.2 Embedding Text for Vector Representation

In natural language processing, representing words and sentences as numerical vectors is a foundational task that enables machines to process and understand human language. Text embeddings are dense vector representations of text that capture semantic meaning and contextual relationships between words or phrases. This approach overcomes the limitations of traditional one-hot encoding, which cannot capture such relationships due to its high dimensionality and sparsity.

One of the earliest and most influential techniques for generating text embeddings is Word2Vec [MCCD13]. This embedding model, through its Skip-Gram and CBOW models, learns embeddings by optimizing the prediction of context words within a given window. This approach captures semantic relationships by leveraging co-occurrence statistics in large text corpora.

Building on this foundation, the GloVe (Global Vectors for Word Representation) model [PSM14] uses matrix factorization techniques to capture global statistical information about word occurrences across a corpus. Unlike Word2Vec, GloVe constructs a global word-word co-occurrence matrix, from which it derives word vectors that reflect the broader semantic relationships within the text.

Recent advancements in text embedding techniques have been driven by the development of contextual embeddings, which consider the context of words within sentences. The ELMo (Embeddings from Language Models) model [PNI+18] generates word representations by utilizing deep, bidirectional language models, allowing embeddings to change based on the context in which words appear. Similarly, BERT (Bidirectional Encoder Representations from Transformers) [DCLT18] uses a transformer-based architecture to produce deeply contextualized embeddings, enabling the model to understand both left and right context simultaneously.

These embedding techniques have revolutionized NLP by providing models with the ability to understand and generate language with enhanced accuracy and nuance. By capturing complex semantic and syntactic relationships, text embeddings serve as a crucial component in a wide range of applications, from sentiment analysis and machine translation to information retrieval and conversational agents.

1.3 The Transformer Architecture

The paper "Attention is All You Need" ([VSP+23]) by Vaswani et al., 2017, introduced the Transformer model. The core innovation of this model is the self-attention mechanism, which allows the model to weigh the relevance of different words in a sequence relative to each other, regardless of their distance. This mechanism operates through the calculation of scaled dot-product attention, where Query, Key, and Value vectors are derived from the input embeddings. These vectors are projected into multiple attention heads, enabling the model to capture various aspects of contextual relationships simultaneously. By applying these attention weights to the Value vectors, the model effectively captures and emphasizes the most contextually relevant parts of the sequence. The result is a model that can process sequences in parallel, significantly improving computational efficiency and allowing for greater scalability in handling longer sequences compared to traditional models.

[VSP+23] originally defined the Attention mechanism as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{1}$$

where Q is the matrix of queries, K is the matrix of keys, V is the matrix of values, and d_k is the dimensionality of the keys (or queries).

The dot product between the query matrix Q and the key matrix K plays a crucial role in determining the similarity between the query and each key. Specifically, the dot product QK^T calculates the similarity between each query vector $\mathbf{q}$ (row of Q) and each key vector $\mathbf{k}$ (row of K).

Mathematically, this operation results in a matrix where each entry $\mathbf{q}_i \cdot \mathbf{k}_j$ represents the similarity score between the i-th query and the j-th key. These scores are then used to weigh the corresponding value vectors in V. This relevance is captured and amplified by the attention mechanism, particularly after applying the softmax function, which normalizes the similarity scores into a probability distribution, emphasizing the most relevant keys.

2 Wiener Filters for Embedded Sequence Comparison

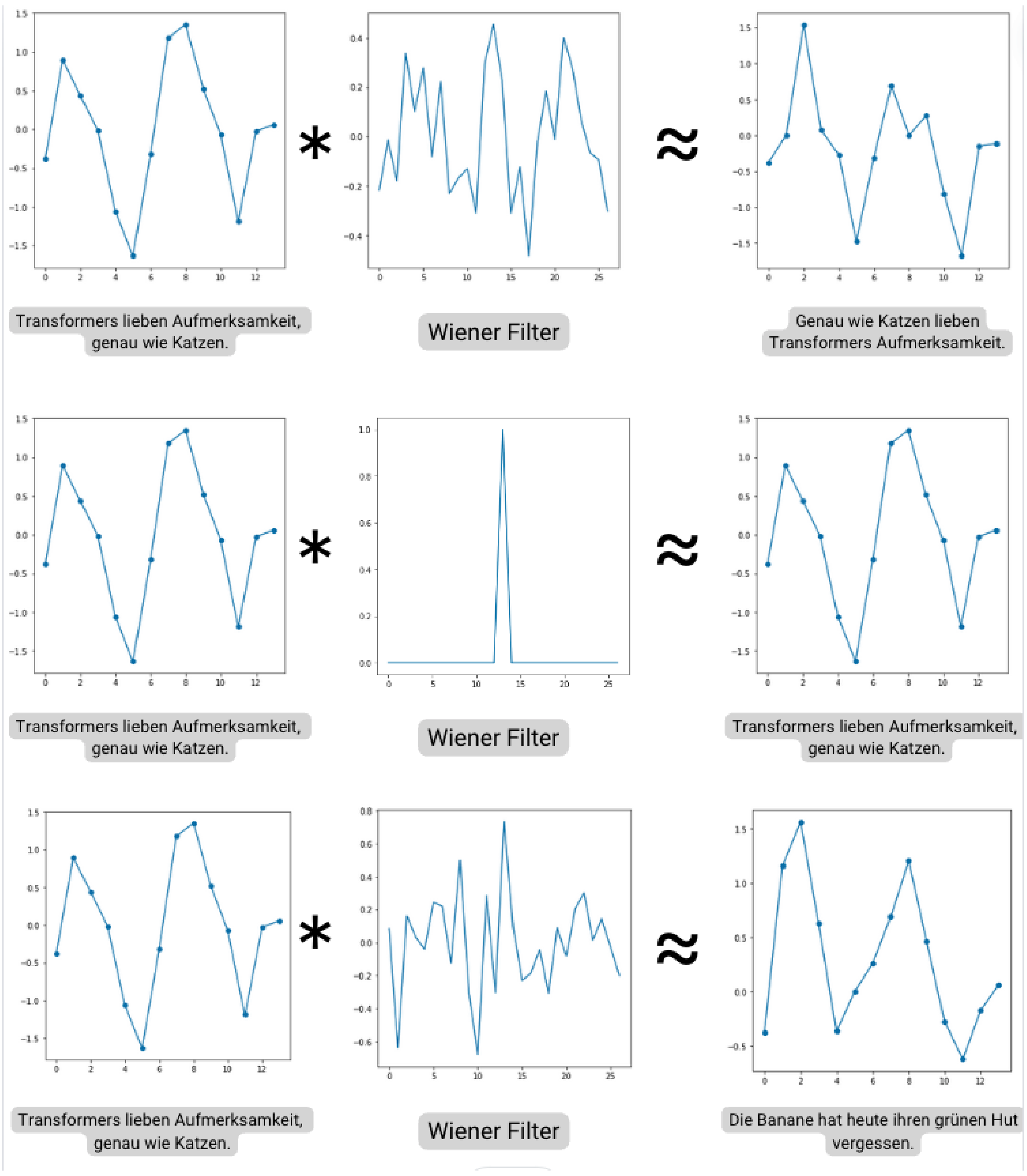


Figure 1: Wiener Filter operations between various embedded text sequences, visualized on the first embedding dimension generated by Word2Vec [MCCD13]. Top: Two similar sentences and their matching Wiener Filter. The magnitudes of the Wiener Filter are low, because the sentences have the same meaning. Middle: Two identical sentences and their matching Wiener Filter. The Wiener Filter matches the Dirac Delta (unit amplitude) because the sentences match perfectly. Bottom: Two completely different sentences and their matching Wiener Filter. The magnitudes of the Wiener Filter are high because of the considerable semantical differences. [CSB+23] produced a similar diagram using images and 2D Wiener Filters.

(The following derivations follow [CSB+23] for cohesiveness and clarity.)
In the frequency domain, the Wiener filter is defined as:

$$v(u,w) = \frac{g^*(u,w)|X(u,w)|^2}{|g(u,w)|^2|X(u,w)|^2 + |n(u,w)|^2}, \tag{2}$$

where $g(u, w)$ is the degradation function, $X(u, w)$ is the Fourier transform of the original image, and $n(u, w)$ is the noise power spectrum. The variables u and w represent the spatial frequency components along the horizontal and vertical axes of the image, respectively.

In this context, $v(u, w)$ is the Wiener filter, which is applied in the frequency domain to restore the original image from its degraded version. The filter works by adjusting the frequency components of the image to counteract the effects of degradation (such as blurring) and noise, aiming to provide the optimal least-squares estimate of the original image. Specifically, the Wiener filter $v(u, w)$ is designed to minimize the mean squared error (MSE) by solving the following optimization problem for v and is then convolved with the noisy image to obtain the least-squares estimate of the original image:

$$\min_v \frac{1}{2} \|Yv - x\|^2 , \tag{3}$$

where v is the Wiener filter, x is a signal and Y is a matrix representing convolution with signal y. For one-dimensional signals, Y is a Toeplitz matrix, while for two-dimensional signals like images, it is a doubly block Toeplitz matrix [Han02]. The solution to this linear least-squares problem is:

$$v(x, y) = (Y^T Y)^{-1} Y^T x, \tag{4}$$

where Y^T denotes the transpose of Y. This solution is efficiently computed in the Fourier domain as:

$$v(x, y) = \mathcal{F}^{-1} \left(\frac{\mathcal{F}\{x\} \otimes \mathcal{F}\{y\} + \lambda}{\mathcal{F}\{x\} \otimes \mathcal{F}\{x\} + \lambda} \right), \tag{5}$$

with $\mathcal{F}$ representing the fast Fourier transform (FFT), $\otimes$ the Hadamard product, and λ a pre-whitening scalar ensuring deconvolution stability. The inclusion of λ in both numerator and denominator ensures that if y is singular (i.e. not invertible), it stabilizes it and mitigates issues related to noise amplification during deconvolution, leading to a more robust and stable estimation of $v(x, y)$ even in the presence of small or zero-valued frequencies in $\mathcal{F}\{x\}$. This regularization is crucial for preventing ill-conditioned results, particularly when dealing with real-world data where perfect inversion is rarely feasible.

The novelty introduced by [CSB$^+$23] is measuring similarity using the MSE between the Wiener filter that matches two signals (images most commonly) and the Dirac Delta (δ), which represents the identity of the convolution operation. This comparison method what the loss and attention mechanisms described in this work rely on to compute a globally-aware similarity.

2.1 From 2D to 1D: adapting Wiener Filters for embedded text comparison

While the Wiener filter's traditional application is in the 2D domain of image processing, its underlying principles have been adapted for 1D sequences before, most commonly for audio signals. The novelty of this work is applying Wiener filters for text embeddings, which are treated as 1D signals. In image processing, the filter minimizes errors by operating across the spatial frequencies in two dimensions (horizontal and vertical). However, in the context of text embeddings, which are 1D sequences, we consider each embedding dimension as analogous to a spatial frequency, where the embedding sequence itself represents the signal.

When applied to text embeddings, the Wiener filter can be utilized to enhance the comparison of these embeddings by reducing the influence of noise or irrelevant features within the embedding space. Just as it filters out noise to enhance image clarity, in text embeddings, it can refine the signal (embedding) by emphasizing the meaningful components that contribute to semantic similarity.

For instance, in dimension-by-dimension sequence comparison, each embedding dimension is treated as a separate signal to which the Wiener filter can be applied. This approach enables the model to focus on the relevant features within each dimension, improving the alignment between the embeddings of different sequences.

This adaptation maintains the core objective of the Wiener filter—minimizing the mean squared error—but reinterprets it within the framework of text embeddings, where the goal is to optimize the similarity measure between sequences or tokens rather than reconstruct an image. By applying Wiener filters in this way, we can leverage their robustness and noise-reduction capabilities to improve the precision and reliability of text sequence comparisons.

The following subsections highlight the potential slicing methods of an embedded text sequence in order to produce signals (1D or 2D) for Wiener filters. Please see Figure 2 for a visualization of the aforementioned slicing methods.

2.2 Dimension-by-Dimension Sequence Comparison

In dimension-by-dimension sequence comparison, the similarity between two sequences of tokens is calculated for each embedding dimension separately. This means that for every embedding dimension, a separate "similarity value" is computed by comparing the corresponding components of the two sequences. Since each embedding dimension represents a different aspect of the token's representation, this approach yields a set of similarity values—one for each dimension. These values can then be averaged across all dimensions to produce a differentiable loss, providing a comprehensive measure of similarity that accounts for the full multidimensional nature of the embedded sequences.

This paper showcases this approach as a regularization loss function using Wiener Filters, applied between the embedded predictions and embedded targets of a translation transformer. By leveraging dimension-by-dimension comparisons, the model can more finely tune the alignment between the predicted and target embeddings, leading to improved translation quality. The experiments section delves deeper into the implementation details and demonstrates how this regularization strategy enhances the model's performance by maintaining a more structured and meaningful embedding space.

2.3 Token-by-Token Embedding Comparison

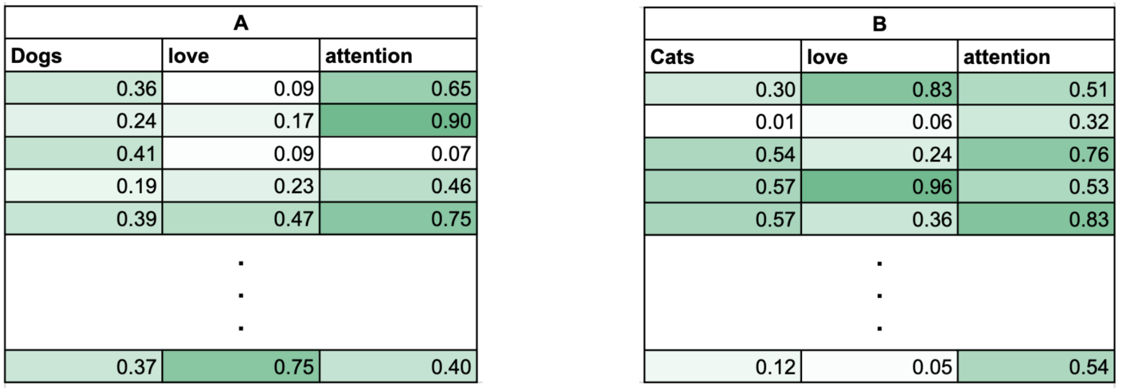
In token-by-token embedding comparison, the similarity between two sequences is calculated by comparing all embedding dimensions of corresponding tokens within the sequences. This approach considers the entire embedding vector for each token, comparing it directly with the embedding vector of the corresponding token in the other sequence. Since the embedding vector captures the token's full representation across all dimensions, this method generates a single "similarity value" for each token pair. Similar to the original Attention mechanism [VSP+23], I arrange these values in a matrix of shape n x n, where n is the sequence length.

This paper showcases this approach in the context of a novel attention mechanism designed for binary classification tasks. By applying token-by-token embedding comparisons, the attention mechanism can more accurately capture and discriminate between subtle differences in token representations that are critical for binary decisions. The applications and experiments section provides further details on how this method is integrated into the model architecture, illustrating its effectiveness in improving classification accuracy by ensuring that the attention mechanism focuses on the most relevant token-level features.

2.4 Two-Dimensional Matrix Comparison

The Two-Dimensional Matrix Comparison approach seeks to integrate the strengths of both dimension-by-dimension and token-by-token comparison methods, offering the most comprehensive measure of similarity between two sequences of embeddings. By simultaneously considering the similarities across both embedding dimensions and token positions, this method creates a two-dimensional matrix where each entry reflects the degree of alignment between specific embedding dimensions and corresponding tokens in the sequences. This matrix can then be analyzed to provide an aggregated similarity score, capturing both the fine-grained dimensional alignments and the overall token-level correspondences.

However, this approach remains theoretical and unexplored within this paper. [CSB+23] found that 2D Wiener Filters don't perform well on "narrow" or "shallow" matrices, only on square or almost square ones. An embedded sequence of length 100 and 512 embedding dimensions would be too narrow for the 2D Wiener filter to meaningfully find context.



A		
Dogs	**love**	**attention**
0.36	0.09	0.65
0.24	0.17	0.90
0.41	0.09	0.07
0.19	0.23	0.46
0.39	0.47	0.75
	⋮	
0.37	0.75	0.40

B		
Cats	**love**	**attention**
0.30	0.83	0.51
0.01	0.06	0.32
0.54	0.24	0.76
0.57	0.96	0.53
0.57	0.36	0.83
	⋮	
0.12	0.05	0.54

Figure 2: The nomenclature used in this paper to refer to different slicing methods is as follows. Take two sequences with their corresponding embeddings as shown in the figure (A and B). Dimension-by-Dimension: comparing $\mathbf{a}_{1i}$ and $\mathbf{b}_{1i}$. Token-by-Token: comparing $\mathbf{a}_{i1}$ and $\mathbf{b}_{i1}$. Two-Dimensional Matrix Comparison: comparing $\mathbf{A}$ and $\mathbf{B}$.

3 Applications and Experiments

In order to test the effectiveness of Wiener filters in the transformer architecture, two experiments have been designed. The first one utilizes an encoder-decoder architecture for translation similar to that of "Attention is All You Need" [VSP+23], but adds a regularization loss using Wiener filters in addition to the Kullback–Leibler divergence loss used in the classification head. The second experiment uses a small BERT model (encoder only), as defined by [DCLT18], for a classification task, with a modified attention mechanism employing Wiener filters.

3.1 Wiener Loss

The concept of Wiener Loss, originally formulated by [CSB+23] for image processing, is now being explored in the context of language transformers. In this novel application, Wiener Loss is utilized as a regularization mechanism applied to the embedding dimensions of target sequences. This approach is designed to enhance the model's robustness by promoting smoother and more consistent representations within the embedding space.

In the context of transformers, the model architecture typically comprises an encoder and a decoder. Inputs are first embedded into a higher-dimensional space before being processed by the encoder, which generates a contextual representation. This encoded information is then fed into the decoder, which, along with the target sequence embeddings, produces the output sequence.

Figure 3 illustrates this training architecture, showing the standard components of the transformer model, such as the encoder, decoder, and softmax output layer. The novelty in this diagram is the introduction of the Wiener Loss, positioned after the decoder and operating on the output embeddings. This loss function works alongside the more conventional Kullback-Leibler Divergence (KLDiv) loss, which compares the predicted output probabilities with the target sequence probabilities. The two losses are automatically weighted [Cen23] and summed to ensure a smooth training process.

By applying Wiener Loss to the output embeddings, the model is encouraged to produce embeddings that are not only accurate in terms of the final output but also smooth and regularized across the embedding dimensions. This can help mitigate issues such as overfitting and instability in the learning process, which are common challenges in high-dimensional spaces. The combined use of KLDiv loss and Wiener Loss represents a dual approach to refining both the accuracy and stability of transformer models, making them more effective and reliable in complex sequence prediction tasks.

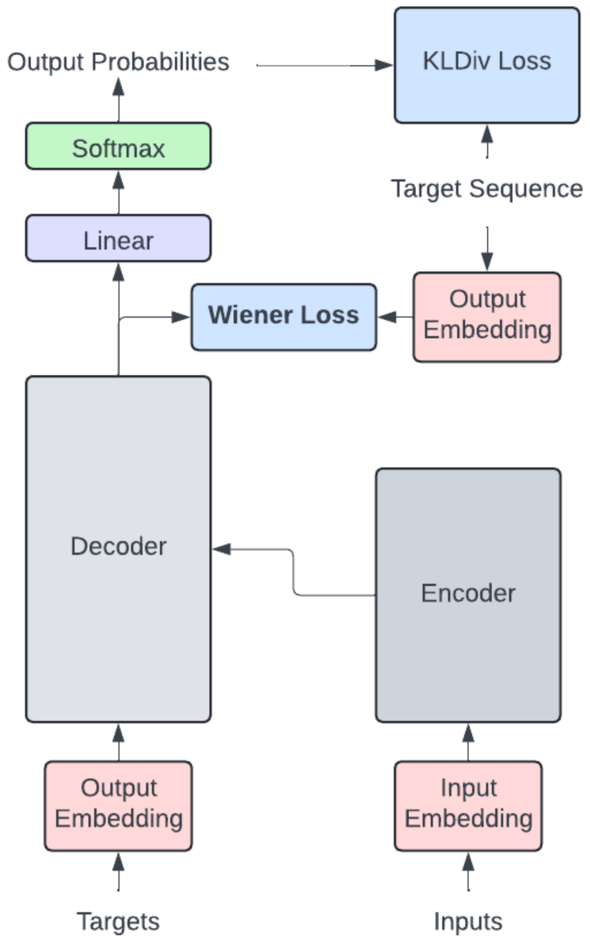


Figure 3: Training diagram of transformer with Wiener Loss. The two losses are automatically weighted [Cen23] and summed to produce a single loss. Modified after [VSP+23].

3.1.1 Formulation

The Wiener Loss is defined by [CSB+23] as the square Mahalanobis distance between the Wiener Filter that, through convolution, matches two signals to each other and the Dirac Delta.

$$\mathcal{L}_W(\mathbf{x}_\theta, \mathbf{y}) = \frac{1}{2} \left\| \mathbf{W} \left\{ \mathbf{v}(\mathbf{x}_\theta, \mathbf{y}) - \delta \right\} \right\|^2, \tag{6}$$

where v is the Wiener Filter that when convolved with model output $\mathbf{x}_\theta$ matches target $\mathbf{y}$, δ is the Dirac Delta (the identity of the convolution operation) and $\mathbf{W}$ is a whitening matrix that aids in regularisation.

In the present example, model output $\mathbf{x}_\theta$ and target $\mathbf{y}$ are embedded sequences of text, separated by embedding dimensions into 1D signals. Therefore, if the pre-classification head model output is identical to the embedded target sequence, the Wiener Filter should be identical to the corresponding Dirac Delta, leading to a zero loss value.

3.1.2 Implementation

In order to test the Wiener Loss, an encoder-decoder Transformer with multi-head attention was used. The model has 6 layers, 8 attention heads per attention layer and model/embedding dimension of 512. Multiple embedding types were implemented and tested, such as Word2Vec [MCCD13] and GloVe [PSM14], alongside the deep-learned embedding mechanism presented in the original Attention paper [VSP+23].

The data used was the same as the original Attention paper [VSP+23] for the English to German translation task - the WMT14 dataset [BBF+14]. This dataset contains approx. 4.5 million sentence pairs, alongside validation and test splits. The data was tokenized using a BytePair encoding algorithm (subword tokenization) [SHB16].

The Transformer architecture and training infrastructure were implemented in PyTorch, based off of *The Annotated Transformer* by [Rus22]. Modifications were made to [Rus22]'s architecture in order to accommodate this experiment and to facilitate multi-GPU training on an NVIDIA DGX server. Furthermore, *The Annotated Transformer* [Rus22] used NLTK tokenization, however that was found to be too slow to build and run, so HuggingFace's *tokenizers* library was used.

Models were trained for 6 or 12 epochs, using 8xA100 GPUs. Introducing the Wiener Loss function had no discernible effect on training time or memory usage.

3.1.3 Results

Evaluation In order to evaluate the translation quality, two metrics were assessed: BLEU Score [PRWZ02] using the NLTK library [BLK09] and SacreBLEU [Pos18] score using the SacreBLEU library [Pos18]. Evaluation was conducted on the test set, which was never used in the training/validation loop.

Control For the control results, three models were trained: (1) a Transformer model with deep-learned embeddings, as described in "Attention is All You Need" [VSP+23]; (2) a similar model using Word2Vec [MCCD13] embeddings trained on the corpus; and (3) a similar model using GloVe [PSM14] embeddings trained on the corpus. The deep-learned embeddings yielded the highest scores, followed by the pre-trained embedding mechanisms, which produced similar results to each other. The control model with deep-learned embeddings achieved BLEU and SacreBLEU scores of 13.69 and 19.35, respectively, for 6 epochs, and 15.53 and 21.45, respectively, for 12 epochs. The Word2Vec and GloVe models showed slightly lower scores, as expected.

Single Stage Training The first attempts to introduce the Wiener Loss were to use it as a secondary loss from the beginning of training. All single-stage models fell short of their control counterparts and, consequently, short of the best performing deep-learned embeddings control model. However, the single-stage deep-learned embeddings model achieved BLEU and SacreBLEU scores of 13.35 and 18.91, respectively, indicating only a slight decrease in performance compared to the control.

Two Stage Training After initial attempts with single-stage training failed to outperform the control models, a two-stage or staggered training approach was employed. This approach involved training the deep-learned embeddings model similarly to the control for a reduced number of epochs (stage one), then continuing training with the inclusion of Wiener Loss (stage two) for the remaining number of epochs. For example, the model was trained normally for three epochs, after which the embedding weights were either frozen or not, and training resumed for an additional three epochs, this time incorporating Wiener Loss. In the control model, the embedding weights were not frozen after the stage split.

Notably, in the case of 6 epochs, the two-stage model where the embedding layers were not frozen produced the best results, surpassing both the frozen variant and the control in BLEU and SacreBLEU evaluations, achieving 14.18 and 20.08, respectively. Notably, even the frozen variant of the two-stage model outperformed the control.

When the training was extended to 12 epochs, the two-stage model with a 25/75 stage split and non-frozen embeddings produced the best results, surpassing the control with BLEU and SacreBLEU scores of 15.63 and 21.62, respectively. The 25/75 model demonstrated superior performance compared to the control up until epoch 8, after which its advantage diminished.

Variant	**Embedding Type**	**Epochs**	**BLEU**	**SacreBLEU**
Control	Learned	6	13.69	19.35
Control	Word2Vec	6	12.14	17.43
Control	GloVe	6	12.06	17.34
Single Stage	Word2Vec	6	11.06	16.34
Single Stage	GloVe	6	11.27	16.53
Single Stage	Learned	6	13.35	18.91
Two Stage (50/50 split)	Learned *(frozen in second stage)*	6	14.15	19.94
Two Stage (33/66 split)	Learned *(frozen in second stage)*	6	14.03	19.89
Two Stage (50/50 split)	Learned *(not frozen)*	6	**14.18**	**20.08**

Table 1: Comparison of BLEU and SacreBLEU scores across different variants and embedding types, all using 6 epochs.

Variant	**Embedding Type**	**Epochs**	**BLEU**	**SacreBLEU**
Control	Learned	12	15.53	21.45
Two Stage (50/50 split)	Learned *(frozen in second stage)*	12	15.01	20.99
Two Stage (25/75 split)	Learned *(frozen in second stage)*	12	14.84	20.73
Two Stage (50/50 split)	Learned *(not frozen)*	12	15.30	21.23
Two Stage (25/75 split)	Learned *(not frozen)*	12	**15.63**	**21.62**

Table 2: Comparison of BLEU and SacreBLEU scores for different variants and embedding types with varying stage splits over 12 epochs

3.1.4 Hyperparameters

λ This parameter, as defined in section 2, controls the numerical stability of the optimization problem in the Fourier domain. Experiments by [CSB+23] found that it exhibits stable

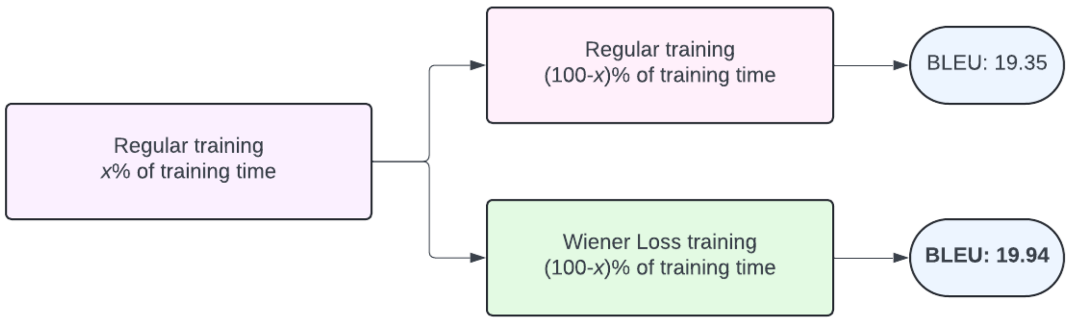


Figure 4: Two stage training of model with SacreBLEU results. In this case, x was equal to 50 and the total number of epochs was equal to 6.

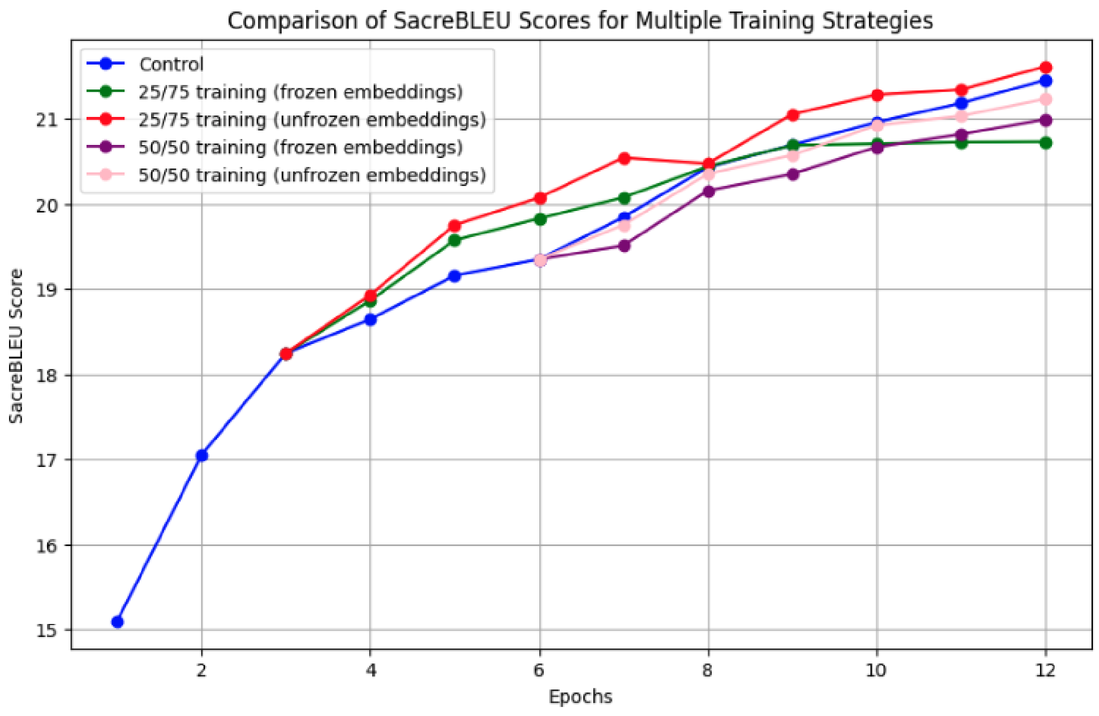


Figure 5: Evolution of SacreBLEU scores for long training. The stage split is where the red and green lines split from the blue line.

behaviour at values between $1e-3 > \lambda > 1e-5$. In the codebase, this hyperparameter is denoted as *epsilon*.

Whitening Matrix This parameter, as defined in section 3.1.1 as $\mathbf{W}$, regularises the Wiener Loss function. It was found to be stable when it contained values in the range of $0.1 < \mathbf{w}_{ij} < 0.3$. It is denoted as *gamma* in the codebase.

Two stage split proportion In two stage training, when to change from normal to Wiener training is an important hyperparameter. In this experiment, 25/75, 33/66 and 50/50 splits were attempted.

3.1.5 Limitations and Future Work

For the Wiener Loss, two main limitations arise from the current work. Firstly, the concept was only tested on a single task (sequence-to-sequence translation) and on a single dataset. To assess whether this regularization loss generalizes, further experiments should be conducted. Secondly, the point at which to switch between the two stages of training is arbitrary. A more systematic and deterministic method is required to identify the optimal point at which to transition to the second stage of training.

Furthermore, more work needs to be dedicated to finetuning the loss parameters λ and Whitening Matrix. Their value was the same throughout all of the training runs, and while the loss exhibited stable behaviour, it is clear that performance can be gained from modifying them. [CSB+23] found that the Wiener Loss was highly sensitive to these hyperparameters, so a logical next step would be to find what value maximises performance.

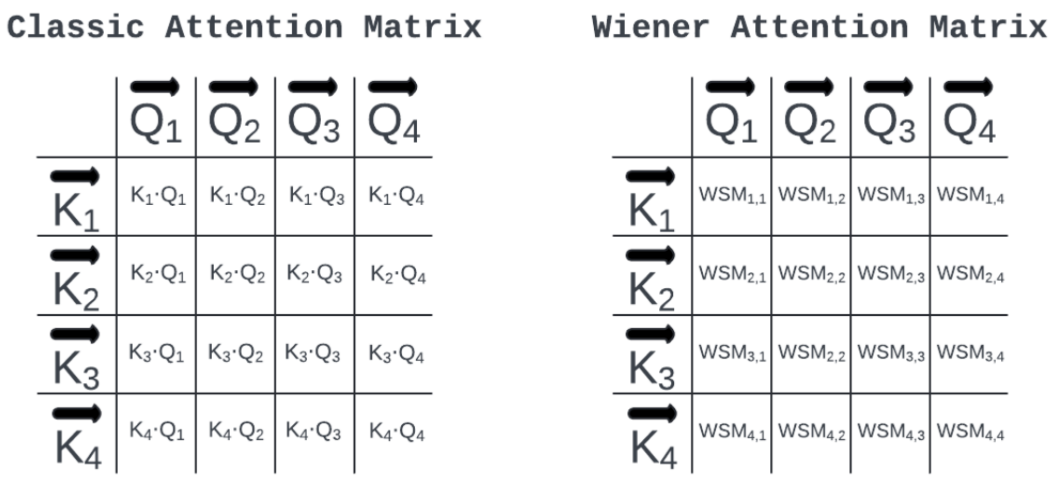
3.2 Wiener Attention

While the Wiener Loss formulated in section 3.1 looks at embedded sequences in a Dimension-by-Dimension perspective, the Wiener Attention mechanism formulated in this section takes a Token-by-Token approach. The former provides a contextual understanding of a text sequence as a whole, highlighting correlations between tokens. The latter compares pairs of tokens across all of their dimensions. As described in section 1.3, the original Attention mechanism by [VSP+23] used the dot-product to compute the similarity. The Wiener Attention replaces the dot product with the Wiener Similarity Metric, with the goal of finding better contextual comparisons between the embeddings of two tokens (or their queries and keys).

3.2.1 Formulation

The Wiener Attention formula is derived from equation (1) and is defined as

$$\text{Wiener Attention}(Q, K, V) = \text{softmin}\left(\frac{WSM(Q, K)}{\sqrt{d_k}}\right) V \tag{7}$$



Classic Attention Matrix

	$\vec{Q}_1$	$\vec{Q}_2$	$\vec{Q}_3$	$\vec{Q}_4$
$\vec{K}_1$	$K_1 \cdot Q_1$	$K_1 \cdot Q_2$	$K_1 \cdot Q_3$	$K_1 \cdot Q_4$
$\vec{K}_2$	$K_2 \cdot Q_1$	$K_2 \cdot Q_2$	$K_2 \cdot Q_3$	$K_2 \cdot Q_4$
$\vec{K}_3$	$K_3 \cdot Q_1$	$K_3 \cdot Q_2$	$K_3 \cdot Q_3$	$K_3 \cdot Q_4$
$\vec{K}_4$	$K_4 \cdot Q_1$	$K_4 \cdot Q_2$	$K_4 \cdot Q_3$	$K_4 \cdot Q_4$

Wiener Attention Matrix

	$\vec{Q}_1$	$\vec{Q}_2$	$\vec{Q}_3$	$\vec{Q}_4$
$\vec{K}_1$	$WSM_{1,1}$	$WSM_{1,2}$	$WSM_{1,3}$	$WSM_{1,4}$
$\vec{K}_2$	$WSM_{2,1}$	$WSM_{2,2}$	$WSM_{2,3}$	$WSM_{2,4}$
$\vec{K}_3$	$WSM_{3,1}$	$WSM_{2,2}$	$WSM_{3,3}$	$WSM_{3,4}$
$\vec{K}_4$	$WSM_{4,1}$	$WSM_{4,2}$	$WSM_{4,3}$	$WSM_{4,4}$

Figure 6: Classic and Wiener Attention Matrices. Classic Attention measures similarity using dot-product, while Wiener Attention uses Wiener Similarity Metric (WSM).

where the scaled dot-product of the original Attention mechanism is replaced by the scaled Wiener Similarity Metric (WSM). The Wiener Similarity Metric is virtually identical to the Wiener Loss described in section 3.1, but instead of providing an average similarity across all samples, it computes a specific similarity measure as a scalar for each pair of query and key vectors. The WSM is lowest when the Wiener Filter that produces the least squares estimate of a token when convolved with another token is equal to the Dirac delta. ‘softmin‘ is used instead of ‘softmax‘ because the Wiener Similarity is defined such that its optimal value is at 0, indicating the highest similarity between vectors.

The Wiener Attention mechanism, applied in a Token-by-Token manner (as described in section 2.2), examines the embedding values within each token as a sequence, enabling the model to capture intricate correlations and patterns across the embedding dimensions. Unlike traditional attention mechanisms, which treat each dimension equally, Wiener Attention can identify and emphasize specific patterns or combinations of embedding values that are most relevant to the task at hand, leading to more informed attention weights. This approach provides a more nuanced understanding of semantic relationships between tokens, particularly in contexts where the relative importance of different embedding dimensions varies depending on the specific tokens being compared. Additionally, the use of Wiener filters introduces robustness to the attention mechanism by minimizing the mean squared error between estimated and true signals, helping the model distinguish between meaningful patterns and noise in the embeddings, which could result in more stable and reliable attention weights.

3.2.2 Implementation

The implementation of the Wiener Attention mechanism is in its early stages, primarily due to computational constraints. Although the computational overhead for a single Wiener Similarity calculation is similar to that of a dot-product, with $T_{\text{dot}} = 2.93 \times 10^{-5}$ seconds and $T_{\text{wiener}} = 8.02 \times 10^{-4}$ seconds for tensors of length 512 on an A10 GPU, the impact becomes significant due to the quadratic scaling of the Transformer's attention mechanism.

In the Transformer architecture, the attention mechanism scales as $O(n^2)$, where n is the number of tokens. For a context window of 200 tokens, the number of similarity operations in a single forward pass is approximately 720,000. While the time difference between T_{dot} and T_{wiener} may seem modest for a single operation, it results in a substantial overhead when scaled across all similarity operations, making Wiener Attention an order of magnitude more time-intensive than classic dot-product attention.

Furthermore, the Wiener mechanism introduces additional computational requirements, such as the Wiener filter, penalty function, and whitening matrix, further increasing overhead and memory usage. Consequently, the inefficiencies necessitated testing the Wiener Attention mechanism in a scaled-down architecture—a single-headed, single-layer BERT [DCLT18] model with a context length of 32 for sentiment analysis on IMDb reviews [MDP+11].

3.2.3 Results

Although the results in Table 3 are promising, as the Wiener Attention models exhibit similar performance to the classic Attention, the visualisation of the Attention matrices in Figure 7 raises questions about the effectiveness of Wiener Filters. As can be seen, the values of the attention probabilities are much closer to each other than those of the classic Attention.

3.2.4 Hyperparameters

λ and $\mathbf{W}$, as described in section 3.1.4, were important hyperparameters for the Wiener Attention mechanism, which influenced the training process. Experiments showed that the same ranges for these hyperparameters applied.

Model Name	Eps	Gamma	Epoch	Precision	Accuracy
Classic	N/A	N/A	2	0.707	**0.707**
Wiener 1	1e-05	0.1	1	**0.710**	0.695
Wiener 2	1e-05	0.2	2	0.692	0.683
Wiener 3	1e-05	0.3	2	0.698	0.697
Wiener 4	1e-04	0.1	3	0.689	0.689
Wiener 5	1e-04	0.2	2	0.698	0.698
Wiener 6	1e-04	0.3	2	0.699	0.695
Wiener 7	1e-03	0.1	1	0.700	0.696
Wiener 8	1e-03	0.2	1	0.695	0.695
Wiener 9	1e-03	0.3	2	0.696	0.685

Table 3: Model Performance with Best Accuracy and Precision

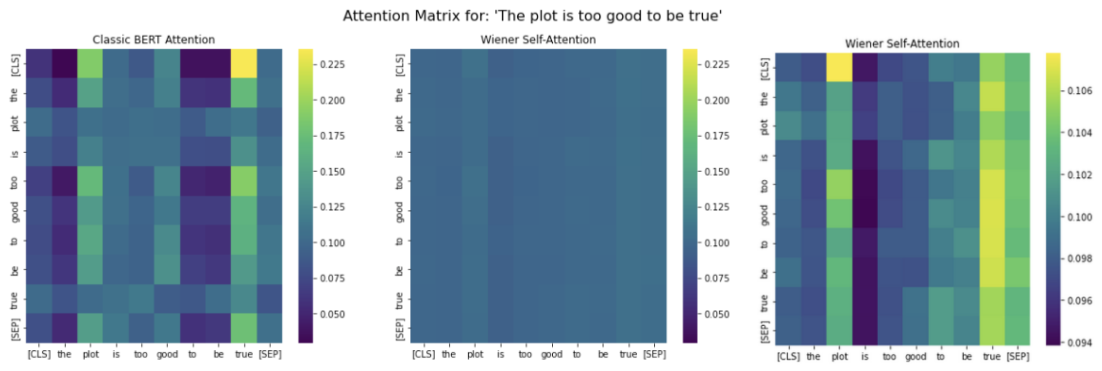


Figure 7: Visualisation of Attention matrices for both Wiener and Classic Attention. The second and third matrices have the same values. The first two matrices are on the same color scale, while the third one has its own.

Sequence Length Another hyperparameter which played an important role was sequence length. In practice, sequence length over 50 would result in out-of-memory issues on an A100 GPU.

3.2.5 Limitations and Future Work

The Wiener Attention remains in its infancy, with its full capabilities yet to be uncovered. Its main limitation at the current time lies in its significant computational overhead compared to classic attention. One possible avenue to solve this would be a FlashAttention [DFE+22] style fused kernel to optimize the operations. Once the overhead is reduced, the Wiener Attention mechanism should be tested on a variety of tasks and datasets.

3.3 Wiener Similarity for RAG

Upon discussions with colleagues, the following idea to implement Wiener filters in RAG [LPP+21] models was created. In RAG, the retrieval component commonly employs cosine similarity to match query embeddings with relevant document embeddings from a large corpus. However, theoretically, the Wiener Similarity Metric, introduced in section 3.2, could replace cosine similarity as the similarity measure used for these retrieval tasks.

The Wiener Similarity Metric offers a more nuanced approach to comparing embeddings by considering the global correlations and signal integrity between token pairs. This could potentially enhance the precision of the retrieval process, allowing for more contextually relevant documents to be identified, particularly in cases where the traditional cosine similarity falls short in capturing deeper semantic relationships.

Though this application of Wiener similarity within RAG has not been implemented in this study, it represents a promising area for future research. Integrating Wiener filters into RAG could offer a new perspective on improving the retrieval phase, potentially leading to more accurate and context-aware responses in generation tasks. Future work should aim to explore this possibility, evaluating the computational feasibility and performance gains of replacing cosine similarity with Wiener similarity in RAG frameworks.

4 Sustainability and Repeatability

All experiments are run using a configuration YAML file, whose path is passed as a system argument to the training loop. This approach ensures that the training loop code almost never has to be modified and that anyone can replicate the experiments, given that they have the same conditions. A configuration file was preferred over simply passing arguments to the command line Python call due to the large number of arguments. Inside of the training loop, the code tries to read training parameters from the configuration file. If it fails to find a certain parameter in the configuration, it falls back to a default value for that parameter.

Test-Driven Development (TDD) and Continuous Integration/Continuous Deployment (CI/CD) practices were integrated into the development process to enhance both the sustainability and repeatability of the experiments. By adhering to TDD principles, each new feature or modification in the codebase was accompanied by a corresponding set of tests. This approach ensured that the training loop and associated utilities were robust and reliable, minimizing the risk of introducing bugs or regressions. Additionally, GitHub Actions was utilized to implement a CI/CD pipeline, automating the process of running tests and validating code changes upon every push or pull request. This not only maintained the integrity of the codebase but also provided immediate feedback to developers, allowing for quick identification and resolution of issues. As a result, the combination of TDD and CI/CD fostered a more stable development environment, ensuring that experiments could be replicated consistently over time, even as the codebase evolved.

To aid any replication attempts, a comprehensive README markdown file was written and attached to the GitHub repository. It contains the information necessary for recreating the experiments, such as setting up the environment, enabling multi-GPU training, writing configuration files and starting the training.

5 Conclusion

The integration of Wiener filters into Transformer models, specifically through the development of Wiener Loss and Wiener Attention mechanisms, represents a significant step forward in enhancing the performance and robustness of Transformer architectures. The experiments conducted in this study demonstrate the potential of these methods to improve the contextual understanding and semantic accuracy of embedded text sequences.

Wiener Loss, when applied as a regularization technique, successfully enhanced the performance of Transformer models, as evidenced by its ability to surpass the control in both BLEU and SacreBLEU scores. This suggests that Wiener Loss is an effective tool for refining the alignment of embeddings, leading to more accurate sequence predictions. The two-stage training approach, in particular, proved to be a pivotal strategy in leveraging the full potential of Wiener Loss, indicating that the timing and method of applying this regularization are critical to its success.

On the other hand, Wiener Attention, though still in its early stages, has shown considerable promise as a novel attention mechanism. Its ability to capture and emphasize the most relevant token-level features offers a new perspective on attention in Transformer models. However, the increased computational complexity associated with Wiener Attention underscores the need for further research and optimization. Future work should focus on reducing the computational overhead and testing the mechanism across a broader range of tasks and datasets.

Additionally, while not explored in this study, the application of Wiener Similarity Metric in Retrieval-Augmented Generation (RAG) models is a promising avenue. Integrating Wiener filters into the retrieval process could potentially enhance the precision of document matching, offering new opportunities for improving retrieval-based NLP tasks.

In this paper, I introduce innovative techniques that bridge signal processing principles with modern NLP, opening up new avenues for model improvement. While Wiener Loss has already proven its effectiveness, Wiener Attention and the potential application in RAG models present exciting frontiers that warrant deeper exploration. The positive results observed here lay the groundwork for future advancements, potentially transforming how we approach text comparison, attention mechanisms, and retrieval processes in NLP models.

Acknowledgements I would like to thank Dr. Lluis Guasch for his continued guidance and mentorship during this project, as well as Deborah Pelacani Cruz for her invaluable direction on Wiener filters. I would also like to thank Prof. Gerard Gorman for his counsel regarding code sustainability and best practices. Furthermore, I am thankful to Oliver Giles for his contribution to the Wiener RAG concept.

References

[BBF+14] Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 workshop on statistical machine translation. In Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, and Lucia Specia, editors, *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.

[BLK09] Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009.

[Cen23] Bai Cenxiao. Strategies for balancing multiple loss functions in deep learning, 2023. Accessed: 2024-08-26.

[CSB+23] Deborah Pelacani Cruz, George Strong, Oscar Bates, Carlos Cueto, Jiashun Yao, and Lluis Guasch. Convolve and conquer: Data comparison with wiener filter, 2023.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Bidirectional encoder representations from transformers. *arXiv preprint arXiv:1810.04805*, 2018.

[DFE+22] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022.

[Han02] Per Christian Hansen. Deconvolution and regularization with toeplitz matrices. *Numerical Algorithms*, 29(4):323–378, 2002.

[LPP+21] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[MDP+11] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[PNI+18] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, 2018.

[Pos18] Matt Post. A call for clarity in reporting BLEU scores. In Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Lucia Specia, Marco Turchi, and Karin Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[Rus22] Sasha Rush. The annotated transformer, 2022. Accessed: 2024-06-11.

[SHB16] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units, 2016.

[VSP$^+$23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[Wie49] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications.* The MIT Press, 1949.