Evaluating Noisy Optimization in Finetuning LMs for Neural Ranking

Daniel Vollmers[1 [0000−0002−5324−4952]], Arnab Sharma[1 [0009−0007−8515−5253]], and Axel-Cyrille Ngonga Ngomo[1 [0000−0001−7112−3516]]

Data Science Group, Heinz Nixdorf Institute, Paderborn University, Germany
{daniel.vollmers, arnab.sharma, axel.ngonga}@uni-paderborn.de
https://dice-research.org/

Abstract. Ranking plays a crucial role in the information retrieval domain by assigning relevance scores to candidate entities based on their similarity to a given mention and context. While LM-based encoder models like BERT have notably improved the quality of entity ranking, optimization challenges, such as convergence issues with adaptive optimizers like Adam, remain largely underexplored. Existing works have shown that injecting noise into an optimizer, such as gradient descent, can serve as a regularizer, leading to improved generalization results. Considering this idea, in this work, we investigate the use of noise injection in the Adam optimization step to improve the optimization process for entity ranking models. By incorporating Gaussian noise into the model parameters, we demonstrate that noise can act as an effective regularizer, helping the model escape saddle points and achieve faster convergence in ranking tasks. More specifically, in this work, we study three ways of injecting noise– (i) uncorrelated, (ii) anticorrelated to previous noise, and (iii) anticorrelated to gradients of the loss function, considering cross entropy as well as pairwise ranking loss. Through extensive evaluation across multiple entity ranking architectures and benchmark datasets, we show that noise-enhanced Adam optimizer improves both convergence speed and model accuracy, offering a promising avenue for enhancing ranking performance. Finally, we evaluate and compare different noise injection approaches, discussing their effectiveness across various models and datasets.

Keywords: Neural Ranking · Retrieval · Optimization.

1 Introduction

Ranking models are an essential component in many NLP application such as information retrieval, entity linking or question answering systems [10,26]. Essentially, the task of a neural ranking model is to assign relevance scores to candidate documents based on their fit to a given query [29]. It involves comparing *context-aware* vector representations of queries with candidate documents, ranking them by similarity. Modern techniques leverage neural encoders like RNNs and transformer-based models (e.g., BERT [6]) to capture complex semantic relationships, offering rich representations. While ranking is a critical step in entity linking tasks, existing research has primarily focused on improving neural architectures [32], leveraging zero-shot learning [27], and

incorporating external features like entity types and priors [9,8,34]. However, less attention is paid to the optimization techniques used during training to finetune the models. More specifically, adaptive optimizers like Adam often face challenges such as local minima and saddle points, particularly in high-dimensional, sparse, and imbalanced datasets, leading to suboptimal performance at inference time [15]. In this work, we study the use of noise injection into the model's parameters, by considering a line of works that aim to increase the generalization performance of the underlying model by doing so [23,12,35,13,18,20,21]. Essentially, our goal is to address saddle point issues and achieve faster convergence towards better-generalized models by applying the noise injection to the parameters in fine-tuning the ranking models.

The role of noise in the optimization step, precisely in Stochastic Gradient Descent (SGD) [25] has been a subject of extensive study, particularly in the context of its impact on generalization [23,12,35,13]. It is widely recognized that the intrinsic stochastic noise in SGD, which arises due to minibatch sampling, tends to guide the optimization process towards flatter minima, which is generally associated with better generalization performance. A recent work by Orvieto et al. [21] proved that noise injection during optimization can lead to better performance in vision models. Although such studies have been explored extensively in the ML domain [18,20], to the best of our knowledge, in information retrieval tasks, and in particular for ranking models, it has not yet been done.

In this work, we study the effect of noise injection into the optimization step to improve the performance of the ranking models while finetuning them for the ranking tasks. More specifically, we add the noise to the parameters of the underlying ranking model after the backpropagation step and during the optimization step. To this end, we, first of all, introduce the formal notion of noise injection into the parameters of the model by considering three different noise injection approaches, (i) uncorrelated noise, (ii) anticorrelated noise based on previously added noise, and (iii) anticorrelated noise using the gradient of the loss function. We present an algorithm that effectively adds noise to the parameters of the models, considering either of the aforementioned approaches, into the optimization step based on the Adam optimizer. To this end, we also show formally that adding Gaussian noise to the parameters of the model in the optimization step, while considering pairwise loss, can work as an explicit regularization.

Apart from theoretical studies and formalizations, we conduct extensive evaluation by applying our noise injection approaches to the Adam optimizer [14] and considering two different types of ranking models, namely bi-encoder [30,31] and cross-encoder [1] models, which are frequently used in entity ranking tasks. Our evaluations suggest that introducing noise in the Adam optimizer can make the training converge much faster and avoid overfitting. Most importantly, we find that noise injection leads to an improvement of performance. Our contribution in this work can be summarized as follows:

- We formalize the notion of three different noise injection approaches in Adam optimizer used in ranking models.
- We introduce the noise injection approaches, considering pairwise ranking loss, specifically adapted to ranking tasks.
- We empirically evaluate several noise injection approaches in the Adam optimizer, considering two encoder models trained on four different ranking datasets.

- We conduct a comparative analysis showing the effectiveness of different noise injection approaches.
- Our code is publicly available[1].

2 Related Work

Ranking models. Recent advancements in NLP have significantly contributed to the ranking tasks. Reimers et al. [24] introduced Sentence-BERT, a model that generates semantically meaningful sentence embeddings using a Siamese network structure, enabling efficient semantic similarity comparison and clustering. The effectiveness of fine-tuning BERT for passage re-ranking, achieving state-of-the-art results on the MS MARCO passage retrieval task has been demonstrated by Nogueira et al. [19]. Wang et al. [30] proposed E5, a family of text embeddings trained through contrastive learning on a large-scale text pair dataset, achieving strong performance in tasks requiring single-vector text representations [30]. More recently, Déjean et al. [5] explored the comparative strengths of cross-encoder rerankers and large language models (LLMs), shedding light on trade-offs between efficiency and ranking effectiveness in different retrieval settings [5]. Similarly, Wu et al. [31] proposed a dense retrieval-based entity-linking approach that improves ranking precision by leveraging transformer-based encoders.

Noise-driven optimization. Adding noise to the gradient descent approach has been widely explored in the literature for its ability to enhance optimization in non-convex settings by introducing noise into the gradient descent process [12,35,28]. For instance, Smith et al. [28] demonstrated that the noise inherent in SGD helps in converging to minima with lower curvature, which would lead to better generalization. A similar outcome has been further observed by Zhang et al. [33], wherein it was shown that SGD's tendency to find flatter minima is linked to its stochastic nature. Moreover, the work of Bradley et al. [4] has established a connection between the level of noise in SGD (influenced by factors like batch size and learning rate) and its ability to find such flat minima, thereby improving the generalization ability of models. Jin et al. [12] demonstrated that escaping saddle points can be achieved by discretizing Langevin dynamics, where noise contributes to diffusion. Zhou et al. [35] further showed that parameter perturbations during optimization help escape spurious local minima in non-convex settings. Adding noise before gradient computation, initially used for smoothing non-smooth convex objectives, has also shown promise in non-convex scenarios by biasing optimization toward flatter minima, which enhances generalization [18,20]. Another approach to introducing noise in the training process involves adding noise to the labels of the training data. Recent studies, such as those by Blanc et al. [3] and HaoChen et al. [11], have demonstrated that label noise can implicitly regularize the learning process by pushing the model to find flatter minima, similar to the effects observed with SGD noise.

Although a vast amount of work has been done exploring the usage of introducing noise to the gradient descent approach, either in parameters or in labels, most of the work has focused on vision-related tasks. The work closest to ours is by Liu et al. [17]

[1] https://github.com/dice-group/RobustRanking

where they introduced PAC-tuning which fine-tunes pretrained models by incorporating PAC-Bayes-driven noise injection into the parameters. In our work, however, we focus on a specific NLP task, namely neural ranking. Herein, we study the effect of noise injection to the Adam optimizer considering both the cross-entropy and pairwise ranking loss. To the best of our knowledge, such study is not yet performed.

3 Neural Ranking Architectures

Bi-encoder rankers use two independent language models to encode the query (the mention context) and the candidates (the entity description) independently into dense vector representations. Next, the dot product of the query and candidate embeddings is computed to score the document representations against the query, which outputs the final probability $\hat{y} \in [0, 1]$, indicating the likelihood of the candidate being the correct entity. Note that, in our work, we consider two different variants of bi-encoder models. The first variant uses a single encoder model to encode both the query and the documents. The weights therein are initialized by the E5 model [30]. We use the term **E5** for this variant. The second one applies two encoder models, one for queries and one for documents, similar to the work of Wu et al. [31]. For this, we consider the original BERT implementation [6] and initialized the weights with its parameters. Within this work, we apply the term **dual bi-encoder** for this variant.

Cross-encoder jointly processes the query and candidate entity by concatenating their texts and feeding them into a transformer-based language model [1]. Herein, the textual representation of the query and the candidate are concatenated. Next this concatenated input is passed through a shared *Language Model+Pooling* layer, which encodes the input sequence and computes a pooled representation that captures the contextual interactions between the tokens of both the query and the candidate entity. Afterwards, the pooled output is forwarded to the scoring function, which outputs a scalar prediction $\hat{y} \in [0, 1]$.

4 Methodology

In our work, we insert noise into the model parameters immediately after the backpropagation and into the optimization step during training. Specifically, given a training sample $x^{(i)}$, the model first predicts an output $\hat{y}^{(i)}$, and let us assume the original output is $y^{(i)}$. After computing the loss $L(\hat{y}^{(i)}, y^{(i)}) = \hat{y}^{(i)} - y^{(i)}$, the gradient of the loss is computed, i.e., the back-propagation step is applied. Thereafter, we insert the noise into the parameters. This noise herein is added into the Adam optimizer before updating the parameters using the moment estimates, allowing us to perturb the parameter space. After inserting the noise, the Adam optimizer is applied to update the model parameters. However, this update is now performed considering the noise-injected parameters, thereby allowing the optimizer to explore parameter regions it might not have encountered otherwise. Next, we formalize different noise injection strategies.

In our work, we inject noise by perturbing the parameters of the encoder components—namely, the query and candidate encoders in the dual bi-encoder, the shared language model in E5. For cross-encoder model, we consider the parameters of both

Algorithm 1: Training loop with noise injection for Document Ranking

Input: Initial parameters θ_l, θ_s; Training data X, Y; Learning rates ξ_l, ξ_s; Number of epochs `num_epochs`
Output: Updated parameters θ_l and θ_s after training
Initialize : Parameters θ_l and θ_s
for *each epoch* **do**
  for *each training sample* $(x^{(i)}, y^{(i)}) \in (X, Y)$ **do**
    Get Prediction for $x^{(i)}$: $\hat{y}^{(i)} = \Psi(x^{(i)}; \theta)$;
    Compute loss: $L(\hat{y}^{(i)}, y^{(i)}) = \hat{y}^{(i)} - y^{(i)}$;
    Calculate the gradient of the loss (Back-propagation):;
      $\nabla\theta(L) = \frac{\partial L(\hat{y}^{(i)}, y^{(i)})}{\partial \theta}$;
    Sample Gaussian noise: $\epsilon \sim \mathcal{G}(0, \mathbf{I})$;
    Add noise to parameters:;
    $\theta \leftarrow \theta - \xi \cdot (\nabla\theta(L) + \delta)$;
    Update parameters using Adam
return θ_l, θ_s *(final learned parameters)*

the language model and the scoring function. To this end, we consider the parameters of the encoder as θ[2]. Herein, we consider three different types of noise injection approaches: (i) uncorrelated noise, (ii) anticorrelated noise using the previous term, and (iii) anticorrelated noise using gradient. Below we discuss them.

Uncorrelated Noise Injection In this case, standard Gaussian distribution is used to generate noise with mean 0 and standard deviation 1 (which in this case is an identity matrix $\mathbf{I}$), therefore, the distribution is defined as $\mathcal{G}(0, \mathbf{I})$. After generating the noise, it is added to the parameters. Consider the noise generated through Gaussian distribution as ϵ. Formally, $\theta' = \theta + \xi * \epsilon$, where ξ is the learning rate of the encoder. Therefore, the learning rate scales the noise in line with the step sizes in the optimization.

Anticorrelated to Previous Noise Unlike standard Gaussian noise, which is independent across iterations, anticorrelated noise integrates historical noise information, allowing for more advanced perturbations during training. Formally, $\theta'_l = \theta + \xi \cdot (-\alpha \cdot \epsilon^{t-1} + \epsilon^t)$, where $\alpha = \frac{\alpha_0}{1+\|\nabla_\theta(L)\|}$ dynamically adjusts based on the gradient norm, $\epsilon^t \sim \mathcal{G}(0, \mathbf{I})$ is the independent Gaussian noise term at iteration t, and ϵ^{t-1} is the noise from the previous iteration. The hyperparameter α controls the influence of past noise. By modulating the impact of noise based on previous iterations, this method can help the optimizer avoid sharp minima and promoting better generalization.

Anticorrelated to Gradient In this method, the noise is directly anticorrelated to the gradient of the loss function, acting as an adaptive regularizer. Formally, $\theta' = \theta + \xi \cdot (-\beta \cdot \nabla_\theta(L) + \epsilon)$, where $\beta = \frac{\beta_0}{1+\|\nabla_\theta(L)\|}$ dynamically adjusts the anticorrelation strength based on the gradient norm, and β_0 is a hyperparameter controlling the overall scale of anticorrelated noise. The dynamic nature of β introduces an adaptive mechanism that reduces noise when the gradient is large and increases noise when the gradient

[2] Note that, for simplicity, we write language model while describing noise injection mechanism for the rest of the paper.

is small. Specifically, when $\|\nabla_\theta(L)\|$ is large (i.e., during steep descent), β becomes small, minimizing the impact of anticorrelated noise. When $\|\nabla_\theta(L)\|$ is small (i.e., near convergence or in flat regions), β increases, emphasizes anticorrelated noise, promoting exploration and helping the model escape shallow minima that may hinder optimization.
Noise Injection in Training Loop Algorithm 1 outlines our overall noise injection approach to the Adam optimizer. In each epoch, for a given training example $(x^{(\mathbf{i})}, y^{(\mathbf{i})})$, the **loss is computed** from the prediction $\hat{y}^{(i)}$. After that, the **back-propagation** is applied by computing the gradients of the loss. Gaussian noise ϵ is sampled, and based on the chosen approach—uncorrelated or anticorrelated—**noise is added to the gradient** for the encoder as δ. Thereafter, **parameter θ is updated** using the Adam optimizer, incorporating gradients and noise scaled by the adaptively adjusted learning rate ξ. Herein δ depends on the approach, for instance, for uncorrelated noise, $\delta = \epsilon$, while for anticorrelated gradient noise, $\delta = -\beta_t \cdot \nabla_\theta(L) + \epsilon$. After the noise addition is done, the Adam optimizer is applied using the first, and second moment estimates. Note that our noise injection approach does not introduce any further complexity in the optimization step. Specifically, this can be performed during the optimization step directly without introducing an extra step in the training process.

5 Evaluation

Both bi-encoder models, i.e., E5 and dual bi-encoder, rely on BERT [6]. E5 comprises one encoder model for the query and the documents, and is already pretrained on the ranking task. For the second model, which we term as dual bi-encoder, we take an architecture that applies two encoder models, where one is used to encode queries and one for the documents. This model has BERT as a foundational model for both the query and document encoders. For the cross-encoder, we consider BERT as well. All foundational BERT models have the same size, i.e., the number of parameters remains the same. For this, we used the version from Hugging Face[3] as BERT implementation and use in-batch negative sampling [31] during training. We trained these models on a server with 128 GB of RAM and an NVIDIA RTX H100 GPU with 80 GB of RAM. Instead of switching to hard negatives, we indexed all entities into a Faiss index [7] at each half-epoch to extract hard negatives. Furthermore, during training, considering the AIDA, Mintaka, and LC-QuAD datasets, we generated each batch including the documents with high-scoring entities from the first document. Herein, we maintained the in-batch strategy throughout training, with random negatives used in the first half-epoch. We trained the model for ten epochs using the default learning rate and parameters for the dual bi-encoder and E5 models. For the evaluation of the biencoder models, the embeddings generated by the models were indexed as well by applying the Faiss-framework.

For the MS MARCO dataset we implemented a similar strategy. However, instead of applying all documents, for computing the hard negatives, we randomly selected $10,000$ negative documents from the whole corpus each time, when we re-indexed the Faiss-index. For each query in the training batch, we selected up to four negatives from the index plus one positive document. We used two queries per batch and kept the

[3] `https://huggingface.co/docs/transformers/model_doc/bert`, `https://huggingface.co/intfloat/e5-base-v2`

in-batch training strategy for the other datasets as well. Similar to bi-encoder models, we trained the cross-encoder model for 10 epochs with 1000 optimization steps per epoch. Since the main objective of this work is to evaluate the influence of noise in the optimization, we had to finetune and evaluate a huge set of models. For this reason, we decided to evaluate the model on a split of 125,000 randomly selected documents from the whole MS MARCO document corpus.

For evaluating and training cross-encoder models, we used a finetuned model of E5 without noisy optimization to compute negatives. For all the samples, we generated a set of 100 negatives for each sample in both the training and test sets. At training time, we chose 14 negatives for each sample plus one positive to generate a candidate set of 15 for each sample. For evaluation, we used all 100 negatives for each of the samples.

5.1 Results & Discussion

In this work, we aim to investigate whether the noise injection to the Adam optimizer can improve the performance of different types of ranking models in diverse ranking tasks. To thoroughly evaluate the effectiveness of our approach, we formulate the following research questions and provide a detailed discussion for each of them below.

- **RQ1.** Does noise injection help in improving the performance of neural ranking models?
- **RQ2.** Which noise injection strategy among the three performs the best?
- **RQ3.** Can noise injection lead to faster convergence and reduce overfitting?

RQ1. In Table 1 we see the results of noise injection strategies on two different bi-encoder models, E5 and dual bi-encoder and the cross-encoder model across 4 datasets We also compute the statistical significance of the results. This was computed by comparing per-query MRR and Recall scores of each noise-injection variant against the No Noise baseline using a paired two-tailed t-test [16].

Herein, we see that E5 demonstrates consistent performance improvements with noise injection. For instance, considering the cross entropy loss, the anticorrelated noise using gradient (Anti-Grad) yields the best MRR and recall in MS MARCO, Mintaka, and AIDA, and anticorrelated noise using previous noise term (Anti-Prev) achieves the best result on LC-QuAD dataset. We see a similar trend considering the pairwise ranking loss as well, wherein Anti-Prev achieves more statistically significant improvement. Therefore, our results indicate the overall effectiveness of noise injection into the Adam optimizer across diverse datasets for the E5 model. The dual bi-encoder model, which contains more parameters than E5, shows greater sensitivity to noise injection. For instance, we observe that Gaussian noise leads to the best MRR and recall in MS MARCO, and in AIDA, whereas Anti-Grad performs the best on LC-QuAD and Mintaka datasets. The effectiveness of noise injection here also can also be observed considering both the cross-entropy and pairwise ranking loss. Herein, we see that both Anti-Grad and Gaussian noise injection approaches achieve statistically significant results. Cross-encoder models, on the other hand, already achieve very high recalls and MRRs on the LC-QuAD, Mintaka, and AIDA. Therefore, we do not see a substantial improvement over the results after injecting noise into the parameters. However, on the largest dataset,

Table 1: MRR and recall results evaluated on four ranking datasets using the E5, Dual bi-encoder, and cross-encoder models. Bolded values indicate the best performance for each dataset. *Significance* columns report statistical significance against the *No Noise* baseline (*: $p < 0.05$, **: $p < 0.01$, n.s.: not significant). LF is loss function; cross entropy (CE) and pairwise loss (PR).

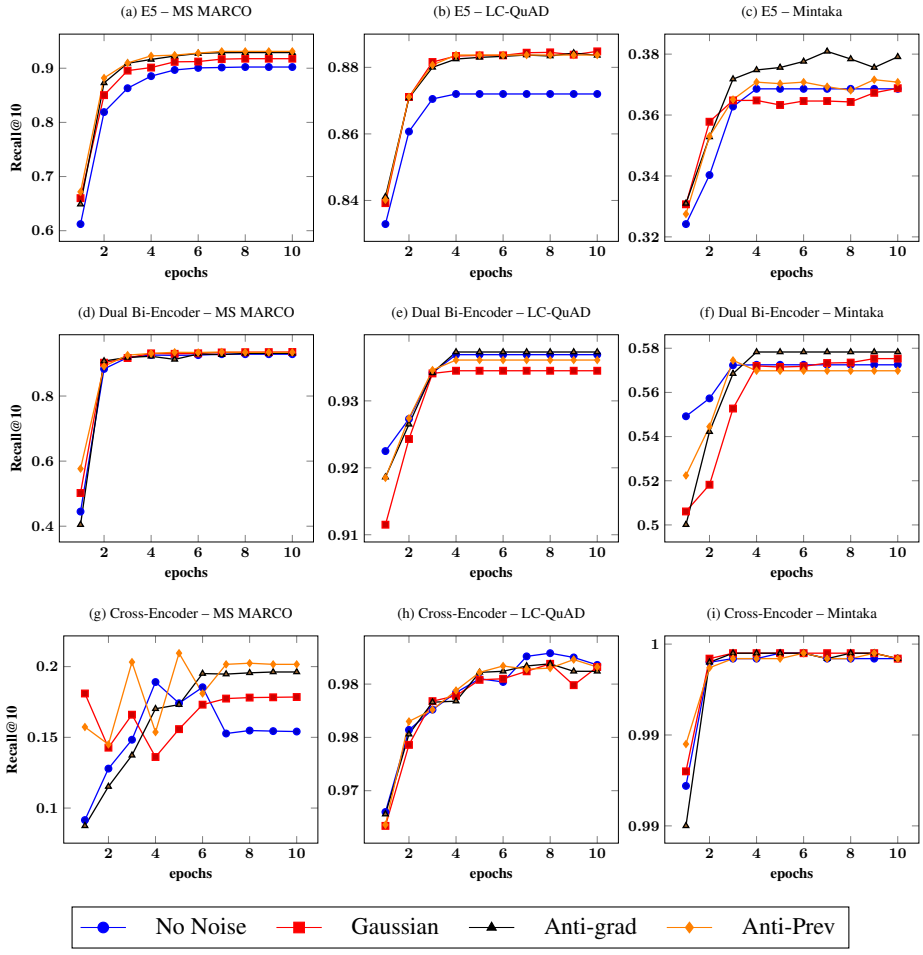
LF	Approach	MRR				Significance	Recall				Significance
		MS MARCO	LC-QuAD	Mintaka	AIDA		MS MARCO	LC-QuAD	Mintaka	AIDA	
E5 Model											
CE	No Noise	0.6423	0.8767	0.2672	0.3049	–	0.8475	0.8719	0.3658	0.1668	–
	Gaussian	0.6327	0.8761	0.2675	0.2971	n.s.	0.8406	0.8725	0.3688	0.1622	n.s.
	Anti-Grad	**0.6708**	0.8972	**0.2704**	**0.3072**	*	**0.8658**	0.8835	**0.3808**	**0.1674**	*
	Anti-Prev	0.6534	**0.8972**	0.2651	0.3034	n.s.	0.8565	**0.8836**	0.3628	0.1638	n.s.
PR	No Noise	0.7456	0.8841	0.2948	0.3009	–	0.8867	0.8888	0.4016	0.1653	–
	Gaussian	0.7566	0.8896	0.2895	**0.3035**	n.s.	0.8675	0.8841	0.3903	0.1651	n.s.
	Anti-Grad	0.7332	0.9000	0.2939	0.3022	*	0.8772	**0.8990**	0.4000	**0.1681**	*
	Anti-Prev	**0.7582**	**0.9056**	**0.3001**	0.3028	**	**0.8852**	0.8943	**0.4109**	0.1668	**
Dual Bi-Encoder Model											
CE	No Noise	0.6479	0.9491	0.4594	0.4414	–	0.8329	0.9369	0.5725	0.2794	–
	Gaussian	**0.6712**	0.9481	0.4625	**0.4635**	*	**0.8529**	0.9363	0.5752	**0.2882**	*
	Anti-Grad	0.6643	**0.9513**	**0.4642**	0.4633	*	0.8458	**0.9372**	**0.5783**	0.2832	*
	Anti-Prev	0.6555	0.9500	0.4496	0.4553	n.s.	0.8406	0.9361	0.5745	0.2828	n.s.
PR	No Noise	0.8001	0.9401	0.4771	0.4615	–	0.9019	0.9388	0.5898	0.3001	–
	Gaussian	**0.8331**	0.9432	0.4955	0.4866	**	0.9223	0.9388	0.5883	0.3112	*
	Anti-Grad	0.8305	**0.9572**	**0.4994**	**0.4872**	**	0.9189	0.9388	**0.5993**	0.3106	*
	Anti-Prev	0.8220	0.9571	0.4801	0.4785	*	**0.9208**	**0.9418**	0.5805	**0.3194**	*
Cross-Encoder Model											
CE	No Noise	0.0441	0.9731	0.9362	0.9941	–	0.1541	**0.9818**	**0.9992**	0.9174	–
	Gaussian	0.0532	0.9720	**0.9392**	0.9937	*	0.1785	0.9816	**0.9992**	0.9168	n.s.
	Anti-Grad	**0.0643**	0.9739	0.9368	0.9938	**	0.1962	0.9812	**0.9992**	0.9158	*
	Anti-Prev	0.0614	**0.9749**	0.9364	**0.9956**	**	**0.2017**	0.9816	**0.9992**	**0.9183**	**
PR	No Noise	0.0891	0.9881	0.9415	**0.9993**	–	0.1672	0.9900	0.9994	0.9245	–
	Gaussian	**0.0922**	0.9817	**0.9511**	0.9944	*	0.1809	**0.9922**	0.9992	**0.9308**	*
	Anti-Grad	0.0800	0.9800	0.9416	0.9941	n.s.	0.2005	0.9876	0.9992	0.9215	*
	Anti-Prev	0.0777	**0.9818**	0.9413	0.9961	n.s.	**0.2173**	0.9887	**0.9995**	0.9201	**

i.e., on MS MARCO, the effect of noise becomes quite prominent. Here we see that noise injection leads to almost **~5%** improvement of recall over no noise setting. This trend is observed considering both the cross-entropy and pairwise ranking loss. Thus, the findings have led us to the following observation:

The effectiveness of noise injection depends on the ranking model and the dataset on which it is trained.

To investigate this further, consider the largest dataset, MS MARCO, wherein we see **~2-3%** gain of performance for the bi-encoder models, and almost **~5%** gain for the cross-encoder is achieved. This demonstrates that the effect of noise in the Adam optimizer shows better generalization performance on larger datasets. Furthermore, we observe that the model size, in terms of the number of parameters, also has an effect on the noise injection. Particularly, we see that the improvement in generalization for the dual bi-encoder, which has more parameters than E5, is better. Then, considering the cross-encoder model the results show substantial improvement by using noise injection into the Adam optimizer. To this end, we observe that on the MS MARCO dataset on which the cross-encoder performs relatively worse, noise injection can lead to almost

Fig. 1: Recall@10 over epochs across 3 datasets for E5, dual bi-encoder, and cross-encoder under four noise injection strategies. Panels (a)–(c): E5; (d)–(f): Dual Bi-Encoder; (g)–(i): Cross-Encoder.



~5% gain of performance. Note that on the other datasets like AIDA, Mintaka, and LC-QuAD the noise injection does not increase the performance significantly. Nonetheless, on those datasets, the cross-encoder model already achieves very high recalls and MRRs, therefore, the room for further improvement through noise injection is limited. This suggests a *diminishing return* effect, where performance gains from regularization techniques like noise injection are more prominent when the *base* model underperforms. In contrast, when the model already achieves near-optimal performance, additional regularization contributes marginal benefits, as also observed by Arpit et al. [2] and Patrini et al. [22]. Overall, our results indicate:

Models like dual bi-encoders and cross-encoders, benefit more from noise injection only when their baseline performance has the scope for generalization improvement, otherwise noise injection does not lead to any substantial improvement.

RQ2. Considering Table 1, we observe that the anticorrelated noise injection approaches perform better than the Gaussian noise injection approach. Specifically, Anti-Prev often shows better statistically significant results considering both the cross-entropy and pairwise ranking loss. However, Anti-Grad also performs on par with Anti-Prev, mostly effective when considering the cross-entropy loss.

More specifically, Anti-Prev leverages the noise added in the previous iteration to inject a new noise term that is anticorrelated. This introduces a form of temporal regularization, creating stable noise injection during training. Unlike independent Gaussian noise or gradient-based noise, this can help in maintaining smooth optimization trajectories, especially in overparameterized models like cross-encoders. Therefore, we observe that in his model, Anti-Prev consistently yields statistically significant results compared to other approaches, considering the MS MARCO dataset. This further indicates that Anti-Prev is particularly effective when the model has already learned strong representations and only needs regularized fine-tuning to avoid overfitting.

On the other hand, the effectiveness of Anti-Grad stems from its dynamic adaptation to the optimization landscape. In particular, this can be attributed to the ability to adjust noise based on the loss by (i) reducing noise during steep descents where the gradient magnitude is large, ensuring that the optimizer follows the correct direction for faster convergence, and (ii) increasing noise in flatter regions or near convergence, which helps the model escape saddle points or shallow local minima that might otherwise trap traditional optimizers. Therefore, such adaptability makes Anti-Grad performing better than injecting uncorrelated noise. Considering these outcomes, we observe the following:

Anticorrelated noise injection approaches, i.e., Anti-Grad and Anti-Prev perform better than the uncorrelated noise injection approaches, considering both the cross-entropy and pairwise ranking loss.

RQ3. Figure 1 shows the stability and convergence behavior of noise injection strategies for E5, dual bi-encoder, and cross-encoder models[4]. Among the bi-encoder models, E5 exhibits increased stability and faster convergence when noise is introduced, particularly with the Anti-Grad strategy, whereas dual bi-encoder, with its greater model complexity, shows varying sensitivity to different noise types. Considering the cross-encoder model, we observe that noise injection substantially leads to faster convergence and moreover, can help with the overfitting issue. More specifically, for E5 model, trained on the largest dataset MS MARCO, we observe that Anti-Grad noise leads to rapid and steady convergence, achieving peak performance at epoch 5 with minimal fluctuations thereafter. In the dual bi-encoder, however, Gaussian noise provides the most stable performance for MS MARCO. On the other hand, without noise, the training requires at least 8-9 epochs to reach the best result for both the ranking models. To this end, we find that the cross-encoder benefited considerably when noise injection is performed. More specifically, as can be seen in Figure 1a, when trained on MS MARCO dataset

[4] Note that, when using pairwise ranking loss, we observe a similar trend in the result. Therefore, due to brevity, we only report the results considering cross-entropy loss.

on which cross-encoder model achieves better recalls as a result of noise injection, also converges faster with Anti-Prev approach. Most importantly, we see that the noise injection approaches can mitigate the overfitting issue which is predominant when using no noise approach. Specifically, we see that after the sixth epoch, the recall value drops considerably when no noise injection approach is used, however, when used, the model stabilizes and does not overfit.

Considering LC-QuAD dataset, both Anti-Grad and Anti-Prev maintain stable recall and MRRs after epoch 4, indicating that anticorrelated noise helps the model converge faster in both the bi-encoder models. Apart from these two large ranking datasets, we see a similar trend also for the Mintaka, and AIDA datasets as well. For both of these 2 datasets, Anti-Grad outperforms all the other approaches by converging within 4-5 epochs in both E5 and bi-encoder. Note that, since the cross-encoder already achieves the highest recalls, as discussed before, the performance gain in terms of faster convergence is not substantial. Nonetheless, our results suggest that adaptive noise strategy like Anti-Grad is particularly effective in datasets with moderate complexity, providing both performance boosts and faster convergence. Overall, based on our results, we can say:

For the bi-encoder models, noise injection accelerates early-stage convergence, with noticeable gains in recall across all the datasets. For the cross-encoder model, the noise injection not only leads to much higher recall and faster convergence, also it helps to reduce the overfitting issue.

Table 2: Best noise injection approach per dataset–model combination

Model	**MS MARCO**	**LC-QuAD**	**Mintaka**	**AIDA**
E5	Anti-Prev	Anti-Prev	Anti-Prev	Anti-Grad
Bi-Encoder	Gaussian	Anti-Grad	Anti-Grad	Anti-Grad
Cross-Encoder	Gaussian	Anti-Prev	Gaussian	Anti-Prev

Optimal Noise Injection per Model and Dataset. When aggregating results across loss functions, we find that different noise injection approaches work differently for models and datasets. For the E5 model, Anti-Prev consistently yields the highest MRR on MS MARCO , LC-QuAD, and Mintaka, suggesting that stabilizing updates across training steps helps this architecture maintain ranking consistency, more specifically in structured or moderately noisy datasets. However, in the case of AIDA, Anti-Grad performs best, likely due to its ability to actively counteract misleading gradients in entity-centric retrieval. In the bi-encoder setting, Gaussian noise is optimal for MS MARCO, indicating that noise injection can indeed help avoid overfitting in large, diverse datasets, whereas Anti-Grad dominates LC-QuAD, Mintaka, and AIDA, reflecting the benefits of gradient correction in smaller, more structured collections. For the cross-encoder, Gaussian noise performs better on MS MARCO and Mintaka, highlighting the usefulness of randomness in large-scale pairwise token matching, while Anti-Prev is superior on LC-QuAD and AIDA, where stable, incremental parameter updates appear to better preserve fine-grained semantic alignment. Table 2 shows the overall results considering which noise injection approach works best on which models and datasets.

6 Conclusion & Future Directions

In this work, we have studied the use of noise injection in the Adam optimizer to improve the performance and convergence speed of ranking models. By incorporating the noise into model parameters during training using different strategies, we demonstrated that noise could act as an effective regularizer, helping the model escape saddle points and converge to flatter minima more effectively. Our experimental results across four different datasets from different domains and two different ranking frameworks (bi-encoder and cross-encoder) consistently showed that noise-enhanced Adam optimization leads to faster convergence and mitigates the overfitting problem. These findings indicate that noise injection is a promising strategy for enhancing the training of ranking models in the domain of information retrieval.

While our work has provided valuable insights into the benefits of noise injection in ranking, we believe there are several possible research directions yet to be explored. For instance, using an adaptive strategy the level of noise and the hyperparameters (related to the anticorrelated approaches) can be adjusted. More specifically, a Bayesian learning strategy could be used therein to select the optimal noise injection parameters. Moreover, the addition of noise could be further studied to find out an optimal bound in terms of model complexity and dataset size beyond which the effect might diminish. Therefore, we envisage that a notion of balanced perturbation needs to be further studied, which can help to determine optimal noise levels and methods for dynamically adjusting perturbations based on training dynamics, loss curvature, model metrics, and datasets. Finally, we envisage exploring the role of curriculum-based noise scheduling, where noise intensity is adapted to the model's confidence. Additionally, integrating such techniques considering contrastive learning objectives may offer further improvements.

Acknowledgements

This work has been supported by the Ministry of Culture and Science of North Rhine-Westphalia (MKW NRW) within the project SAIL under the grant no NW21-059D, the project WHALE (LFN 1-04) funded under the Lamarr Fellow Network programme by the Ministry of Culture and Science of North Rhine-Westphalia (MKW NRW), the European Union's Horizon Europe research and innovation programme under grant agreement No 101070305, and by the German Federal Ministry of Research, Technology and Space (BMFTR) within the project KI-OWL under the grant no 01IS24057B.

References

1. Agarwal, O., Bikel, D.M.: Entity linking via dual and cross-attention encoders. CoRR **abs/2004.03555** (2020), https://arxiv.org/abs/2004.03555
2. Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M.S., Maharaj, T., Fischer, A., Courville, A.C., Bengio, Y., Lacoste-Julien, S.: A closer look at memorization in deep networks. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Proceedings of Machine Learning Research, vol. 70, pp. 233–242. PMLR (2017), http://proceedings.mlr.press/v70/arpit17a.html

3. Blanc, G., Gupta, N., Valiant, G., Valiant, P.: Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In: Conference on Learning Theory, COLT 2020. Proceedings of Machine Learning Research, vol. 125, pp. 483–513. PMLR (2020), http://proceedings.mlr.press/v125/blanc20a.html
4. Bradley, A.V., Gomez-Uribe, C.A.: How can increased randomness in stochastic gradient descent improve generalization? CoRR **abs/2108.09507** (2021), https://arxiv.org/abs/2108.09507
5. Déjean, H., Clinchant, S., Formal, T.: A thorough comparison of cross-encoders and llms for reranking SPLADE. CoRR **abs/2403.10407** (2024). https://doi.org/10.48550/ARXIV.2403.10407, https://doi.org/10.48550/arXiv.2403.10407
6. Devlin, J., Chang, M., Lee, K., Toutanova, K.: 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics (2019), https://doi.org/10.18653/v1/n19-1423
7. Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.E., Lomeli, M., Hosseini, L., Jégou, H.: The faiss library. IEEE Transactions on Big Data **12**(2), 346–361 (2026). https://doi.org/10.1109/TBDATA.2025.3618474
8. Fang, Z., Cao, Y., Li, Q., Zhang, D., Zhang, Z., Liu, Y.: Joint entity linking with deep reinforcement learning. In: The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019. pp. 438–447. ACM (2019), https://doi.org/10.1145/3308558.3313517
9. Ganea, O., Hofmann, T.: Deep joint entity disambiguation with local neural attention. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017. pp. 2619–2629. Association for Computational Linguistics (2017), https://doi.org/10.18653/v1/d17-1277
10. Guo, J., Fan, Y., Ai, Q., Croft, W.B.: A deep relevance matching model for ad-hoc retrieval. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. p. 55–64. CIKM '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2983323.2983769, https://doi.org/10.1145/2983323.2983769
11. HaoChen, J.Z., Wei, C., Lee, J.D., Ma, T.: Shape matters: Understanding the implicit bias of the noise covariance. In: Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA. Proceedings of Machine Learning Research, vol. 134, pp. 2315–2357. PMLR (2021), http://proceedings.mlr.press/v134/haochen21a.html
12. Jin, C., Ge, R., Netrapalli, P., Kakade, S.M., Jordan, M.I.: How to escape saddle points efficiently. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning, ICML. Proceedings of Machine Learning Research, vol. 70, pp. 1724–1732. PMLR (2017), http://proceedings.mlr.press/v70/jin17a.html
13. Jin, C., Netrapalli, P., Ge, R., Kakade, S.M., Jordan, M.I.: On nonconvex optimization for machine learning: Gradients, stochasticity, and saddle points. J. ACM **68**(2) (2021), https://doi.org/10.1145/3418526
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
15. Kunstner, F., Milligan, A., Yadav, R., Schmidt, M., Bietti, A.: Heavy-tailed class imbalance and why adam outperforms gradient descent on language models. In: Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J.M., Zhang, C. (eds.) Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 -

15, 2024 (2024), http://papers.nips.cc/paper_files/paper/2024/hash/350e718ff74062b4bac2c6ffd9e1ac66-Abstract-Conference.html
16. Lehmann, E.L.: Introduction to Student (1908) The Probable Error of a Mean, pp. 29–32. Springer New York, New York, NY (1992). https://doi.org/10.1007/978-1-4612-4380-9_3
17. Liu, G., Xue, Z., Zhang, X., Johnson, K.M., Wang, R.: Pac-tuning: Fine-tuning pre-trained language models with pac-driven perturbed gradient descent. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP. pp. 12178–12189. Association for Computational Linguistics (2023), https://doi.org/10.18653/v1/2023.emnlp-main.748
18. Liu, T., Li, Y., Wei, S., Zhou, E., Zhao, T.: Noisy gradient descent converges to flat minima for nonconvex matrix factorization. In: The 24th International Conference on Artificial Intelligence and Statistics, AISTATS. Proceedings of Machine Learning Research, vol. 130, pp. 1891–1899. PMLR (2021), http://proceedings.mlr.press/v130/liu21e.html
19. Nogueira, R.F., Cho, K.: Passage re-ranking with BERT. CoRR **abs/1901.04085** (2019), http://arxiv.org/abs/1901.04085
20. Orvieto, A., Kersting, H., Proske, F., Bach, F.R., Lucchi, A.: Anticorrelated noise injection for improved generalization. In: International Conference on Machine Learning, ICML. Proceedings of Machine Learning Research, vol. 162, pp. 17094–17116. PMLR (2022), https://proceedings.mlr.press/v162/orvieto22a.html
21. Orvieto, A., Raj, A., Kersting, H., Bach, F.R.: Explicit regularization in overparametrized models via noise injection. In: Ruiz, F.J.R., Dy, J.G., van de Meent, J. (eds.) International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain. Proceedings of Machine Learning Research, vol. 206, pp. 7265–7287. PMLR (2023), https://proceedings.mlr.press/v206/orvieto23a.html
22. Patrini, G., Rozza, A., Menon, A.K., Nock, R., Qu, L.: Making deep neural networks robust to label noise: A loss correction approach. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 2233–2241. IEEE Computer Society (2017). https://doi.org/10.1109/CVPR.2017.240, https://doi.org/10.1109/CVPR.2017.240
23. Polyak, B.T., Tsybakov, A.B.: Optimal order of accuracy of search algorithms in stochastic optimization. Problemy Peredachi Informatsii **26**(2), 45–53 (1990)
24. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019. pp. 3980–3990. Association for Computational Linguistics (2019), https://doi.org/10.18653/v1/D19-1410
25. Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics pp. 400–407 (1951)
26. Shoshan, E., Radinsky, K.: Latent entities extraction: How to extract entities that do not appear in the text? In: Korhonen, A., Titov, I. (eds.) Proceedings of the 22nd Conference on Computational Natural Language Learning. pp. 200–210. Association for Computational Linguistics, Brussels, Belgium (Oct 2018). https://doi.org/10.18653/v1/K18-1020, https://aclanthology.org/K18-1020/
27. Sil, A., Kundu, G., Florian, R., Hamza, W.: Neural cross-lingual entity linking. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA,

February 2-7, 2018. pp. 5464–5472. AAAI Press (2018), https://doi.org/10.1609/aaai.v32i1.11964
28. Smith, S.L., Elsen, E., De, S.: On the generalization benefit of noise in stochastic gradient descent. In: Proceedings of the 37th International Conference on Machine Learning, ICML 2020. Proceedings of Machine Learning Research (2020), http://proceedings.mlr.press/v119/smith20a.html
29. Trabelsi, M., Chen, Z., Davison, B.D., Heflin, J.: Neural ranking models for document retrieval. Information Retrieval Journal **24**(6), 400–444 (Oct 2021). https://doi.org/10.1007/s10791-021-09398-0
30. Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., Wei, F.: Text embeddings by weakly-supervised contrastive pre-training. CoRR **abs/2212.03533** (2022), https://doi.org/10.48550/arXiv.2212.03533
31. Wu, L., Petroni, F., Josifoski, M., Riedel, S., Zettlemoyer, L.: Scalable zero-shot entity linking with dense entity retrieval. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020. pp. 6397–6407. Association for Computational Linguistics (2020), https://doi.org/10.18653/v1/2020.emnlp-main.519
32. Yamada, I., Asai, A., Shindo, H., Takeda, H., Matsumoto, Y.: LUKE: deep contextualized entity representations with entity-aware self-attention. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020. pp. 6442–6454. Association for Computational Linguistics (2020), https://doi.org/10.18653/v1/2020.emnlp-main.523
33. Zhang, G., Li, L., Nado, Z., Martens, J., Sachdeva, S., Dahl, G.E., Shallue, C.J.: Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS 2019 (2019), https://dl.acm.org/doi/abs/10.5555/3454287.3455023
34. Zhang, Z., Sind, X., Liu, T., Fang, Z., Li, Q.: Joint entity linking and relation extraction with neural networks for knowledge base population. In: 2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020. pp. 1–8. IEEE (2020), https://doi.org/10.1109/IJCNN48605.2020.9207021
35. Zhou, M., Liu, T., Li, Y., Lin, D., Zhou, E., Zhao, T.: Toward understanding the importance of noise in training neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, ICML. Proceedings of Machine Learning Research, PMLR (2019), http://proceedings.mlr.press/v97/zhou19d.html