ImpactAi at ImageEval 2025 Shared Task: Region-Aware Transformers for Arabic Image Captioning – A Case Study on the Palestinian Narrative

Rabee Al-Qasem
AI Developer, GGateway
Nablus, Palestine
r.qasim@ggateway.tech

Mohannad Hendi
Independent Researcher
Nablus, Palestine
M.hendi1@student.aaup.edu

Abstract

This paper presents our approach to the ImageEval Shared Task for Arabic image captioning, with a focus on the Captioning with Region Features Transformer (CRAFT) model. The system combines Faster R-CNN-based region feature extraction with a custom vision transformer encoder and transformer decoder, trained on a custom, human-annotated dataset with a Palestinian context. To ensure fairness in evaluation, we compare CRAFT with an alternative Vision-Encoder–Decoder system (AraViT-GPT). Performance was assessed using BLEU, ROUGE, cosine similarity, and an LLM-based semantic evaluation. Results show that CRAFT achieved the highest cosine similarity (56.22 on the test set), indicating superior semantic fidelity to reference captions, while AraViT-GPT showed marginally better n-gram precision and LLM-judge scores. These findings demonstrate the advantages of region-focused visual encoding for Arabic caption generation, particularly in the context of context-rich and historically significant imagery.

1 Introduction

This paper presents our work in **Subtask 2 of the ImageEval 2025 Shared Task** on developing and evaluating image captioning models (Bashiti et al., 2025). This subtask focuses on generating culturally relevant and contextually accurate Arabic captions for images.

We developed **CRAFT** (Captioning with Region Features Transformer), which uses region-level visual features extracted via Faster R-CNN (ResNet-50 backbone), followed by a custom vision transformer encoder and transformer decoder. We also compared this main model with a custom transformer-based model, **AraViT-GPT**, as well as the baseline model provided in the shared task.

Our experiments showed that CRAFT achieved the highest semantic fidelity, with cosine similarity scores of 57.22 (validation) and 56.22 (test), while AraViT-GPT slightly outperformed in n-gram precision and LLM judge scores. In the official leaderboard, our system ranked 4th in both cosine similarity and LLM-based evaluation, and 5th in the human evaluation track, where real annotators assessed caption quality.

The main challenge we encountered was named entity recognition, where the model occasionally produced factual inaccuracies when identifying specific people or locations, despite correctly recognizing the general scene context. Our code and training pipeline are available at Github .

2 Background

The main task addressed in this paper is image captioning (Subtask 2), where the model takes an image as input and generates an Arabic caption for it. The model's performance is then evaluated using metrics such as BLEU, ROUGE, cosine similarity, and an LLM-based semantic scoring metric.

The provided dataset consists of 3,471 manually captioned images in Arabic. The dataset encompasses a diverse range of scenes, including buildings, people, and artifacts. It presents various challenges, such as identifying individuals' names and handling both colored and black-and-white images. Each image features a unique Arabic caption, annotated by a human, that provides a detailed description of the image. The dataset particularly focuses on the Palestinian historical narrative.

3 System Overview

Our empirical study compares two algorithms: Captioning with Region Features Transformer (CRAFT) and (AraViT-GPT). In this section, we discuss the CRAFT model, which achieved the highest cosine similarity results of the two models. Details of AraViT-GPT are in Section 6, which we included as part of our ablation study.

Proceedings of The Third Arabic Natural Language Processing Conference, pages 401–407

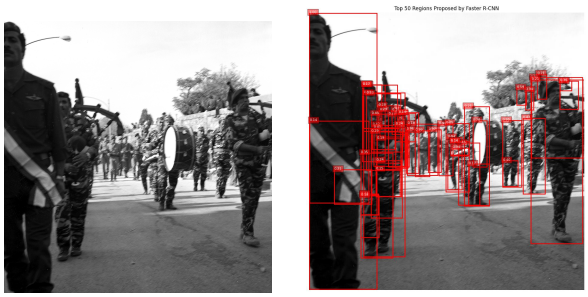


Figure 1: Faster R-CNN output showing region proposals, capturing multiple people and contextual objects.

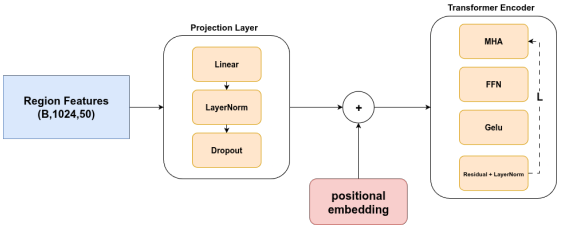


Figure 2: Architecture of the Vision Encoder, showing the three main components: projection of region features, addition of learned positional embeddings, and a multi-layer Transformer encoder.

3.1 Region Feature Extraction using Faster R-CNN

Given the heterogeneity of our images (crowds, many objects, and photo-of-photo artifacts), we use a pre-trained Faster R-CNN (Ren et al., 2015) with a ResNet-50 backbone (He et al., 2016) to propose regions. Rather than process full images, we extract $k = 50$ regions of interest (ROIs) per image, yielding a 50×1024 embedding tensor; these features, together with normalized box coordinates, serve as input to the vision transformer backbone (Fig. 1). We also tested a dynamic k chosen by clustering ROIs via the elbow method (typically 15–70 per image), but it sometimes dropped important objects (Hendi et al., 2023). Also we tried a fixed $k = 100$, which offered no improvement over $k = 50$.

3.2 Visual Encoder

Our custom Vision Encoder processes region features from Faster R-CNN. The features are first unified by a projection layer, combined with learned positional embeddings, and then passed to a Transformer encoder with an optimal configuration of two layers (L=2) and two attention heads (H=2), as determined by hyperparameter tuning using a grid search (Bergstra and Bengio, 2012). This process generates a [Batch, 50, 768] embedding that is subsequently passed to the caption decoder (Fig. 2).

3.3 Caption Decoder

The caption decoder generates Arabic text using ArabGlossBERT (Al-Hajj and Jarrar, 2022; Antoun et al., 2020), which provides a vocabulary of approximately 64,000 tokens. Token and positional embeddings are mapped to a 768-dimensional space to match the visual features from the encoder. Both text embeddings and visual features are then fed into a Transformer decoder. This decoder is configured with a maximum caption length of M=97, with its optimal parameters of L=2 layers and H=2 attention heads determined by a grid search (Fig. 3).

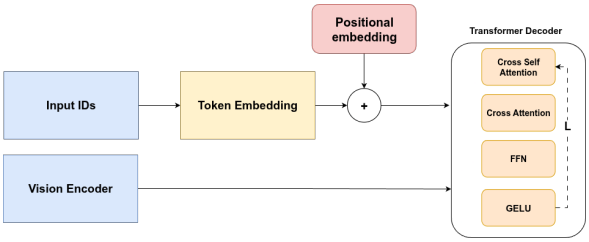


Figure 3: Architecture of the Caption Decoder, showing token embedding, addition of positional embeddings, Transformer decoder, and final projection to vocabulary.

3.4 Sequence Decoder

For caption generation, we use the beam search method (Vaswani et al., 2017). In this iterative process, the decoder maintains a set of candidate sequences (beams) at each step. A causal mask is applied to prevent the model from attending to future positions, and only the top candidates, ranked by their cumulative probabilities, are retained. The process terminates upon reaching the maximum sequence length of 97, at which point the highest-scoring sequence is selected as the final caption.

4 Experimental Setup

4.1 Data Split

All experiments were conducted using the Shared Task dataset, with no external data involved in the process. The dataset was split into training (2,718 samples) and test (753 samples) sets. During the Task, we also received 75 images for validation. The test set was used exclusively to evaluate the models' generalization performance.

4.2 Data Preprocessing

We normalized all Arabic captions to reduce noise: unified orthographic variants (Alef, Yeh, Teh Marbuta), removed Tatweel and diacritics, standardized punctuation to Latin equivalents, and collapsed redundant whitespace. The slash (/) in date-like captions was treated as an unknown token by the tokenizer, so we replaced it with a space (e.g., '09/1970' $\rightarrow$ '09 1970') before tokenization. These steps improved the model's predictions.

4.3 Training setup

Prior to the full training, we conducted hyperparameter tuning over a limited run of 10 epochs. The grid search included the number of encoder and decoder layers, the number of attention heads, and the batch size. For more details on the hyperparameter tuning, see Appendix A.1. The optimal configuration was found to be 2 layers for both the encoder and decoder, 2 attention heads, and a batch size of 8 (see Appendix A.1). This resulted in a model size of approximately 132 million trainable parameters.

Using these settings, the full training was performed on a NVIDIA T4 GPU via Google Colab for 40 epochs with early stopping, which occurred at epoch 30. The AdamW optimizer (Loshchilov and Hutter, 2017) was employed alongside a linear learning rate scheduler initialized at 1×10^{-4}. Cross-entropy loss was selected as the objective function. To further enhance model generalization, we applied online data augmentation where each sample was exposed to randomized transformations on every epoch. These included horizontal flips, mild affine transformations (rotations up to $\pm 15^{\circ}$, translations up to ± 5, scaling between 0.9–1.1, and shear up to $\pm 3^{\circ}$), as well as photometric changes such as brightness and contrast adjustments, gamma correction to simulate aging effects, and the addition of light Gaussian noise with $\sigma \leq 0.05$. Collectively, these augmentations increased sample diversity and made the model more robust to variations in historical images.

Finally, our implementation, developed in Python, utilized PyTorch and TorchVision for model training (Paszke et al., 2019; Marcel and Rodriguez, 2010), alongside NumPy. We used Matplotlib and Seaborn for visualization, Hugging Face Transformers for transformer components, and Weights & Biases (W&B) for experiment tracking.

4.4 Evaluation Metrics

To comprehensively assess both models, we used a mixed set of evaluation metrics provided in the shared task description paper (Bashiti et al., 2025): BLEU (Papineni et al., 2002) for n-gram precision, ROUGE (Lin, 2004) for recall-oriented overlap, cosine similarity for semantic alignment in embedding space, and a Large Language Model (LLM) judge (GPT-4o) to imitate human judgments (Al-Qasem et al., 2025). While BLEU and ROUGE quantify surface overlap, cosine similarity indicates whether a predicted caption conveys the reference meaning. Because semantic similarity is our primary objective, we assign greater weight to cosine similarity when comparing models. This weighting, together with the LLM judge, guided our conclusion about which model best suits the target application.

5 Results

In this section, we report the performance of the two models, provide examples from the best-performing model, and highlight some flaws observed in its predictions. As shown in Table 1, and following our protocol in Section 4, cosine similarity is treated as the *primary* metric because it best captures semantic fidelity to the reference captions.

5.1 Quantitative findings

The results in Table 1 show that the CRAFT model achieves higher scores on both splits in terms of cosine similarity, with 57.22 on the validation set and 56.22 on the test set, indicating greater semantic closeness in the embedding space. On the other hand, AraViT-GPT holds a slight lead in n-gram precision (BLEU-1–4) and achieves the highest LLM-Judge score (26.55 on the test set). Both models record near-zero results on the ROUGE metrics, which in this case reflects the lack of exact lexical overlap between the generated and reference captions. For context, we also benchmarked both models against the shared-task baseline, see Table 2. While both models exceed the baseline on BLEU-1–4, the fine-tuned baseline attains the highest cosine similarity (58.46) and LLM-as-judge score (30.82) on the test set.

5.2 Prediction Discussion

To complement the quantitative results, we present a qualitative comparison between the two models' outputs on selected images from the Test dataset.

Metric	CRAFT		AraViT-GPT	
	Val	Test	Val	Test
BLEU-1	19.68	19.07	**21.05**	**21.40**
BLEU-2	10.56	9.66	**11.48**	**11.59**
BLEU-3	7.01	6.00	**8.54**	**8.48**
BLEU-4	4.21	3.78	**5.18**	**5.22**
ROUGE-1	0	0	0	0
ROUGE-2	0	0	0	0
ROUGE-L	0	0	0	0
Cosine Similarity Mean	**57.22**	**56.22**	55.35	55.46
LLM Judge (/100)	22.07	22.34	**26.07**	**26.55**

Table 1: Comparison of CRAFT and AraViT-GPT performance on validation and testing sets.

	Ours		Baseline (Qwen 2.5-VL 7B)	
Metric	CRAFT (Test)	AraViT-GPT (Test)	Zero-shot	Fine-tuned
BLEU-1	19.07	**21.40**	9.92	16.98
BLEU-2	9.66	**11.59**	3.23	8.62
BLEU-3	6.00	**8.48**	1.90	5.43
BLEU-4	3.78	**5.22**	1.33	3.05
ROUGE-1	0	0	0	0
ROUGE-2	0	0	0	0
ROUGE-L	0	0	0	0
Cosine Similarity Mean	56.22	55.46	55.77	**58.46**
LLM Judge (/100)	22.34	26.55	27.11	**30.82**

Table 2: Test-set comparison between our models and the baseline. Baseline values are taken from the shared-task notebooks and converted to a percentage scale for comparability.

These examples highlight cases where the captions are accurate, partially correct, or fail to capture the main scene, providing deeper insight into each model's strengths and weaknesses.

Example 1 Figure 4 shows a large crowd scene which includes a public demonstration in support of Palestine. The CRAFT caption (**"A photo of a demonstration in Beirut following the events of September 1970."**) uses the correct event term *demonstration* and gives a clear, precise description of the scene, and it also gives a year and location for the image. By contrast, the AraViT-GPT caption (**"An image of part of the Palestinian activities"**) is generic, does not explicitly describe the event, and reads less fluently. CRAFT provides a more accurate and informative caption for this image.

Example 2 Figure 5 shows a group of individuals in military uniforms gathered around the Palestinian leader Yasser Arafat while wearing sunglasses in a training camp. The CRAFT model produced the caption: **"An image of Yasser Arafat, Farouk Qaddoumi, and Ismail Shammout in one of the Palestinian revolution camps"**. This output demonstrates the model's ability to correctly identify Yasser Arafat, who is wearing sunglasses,

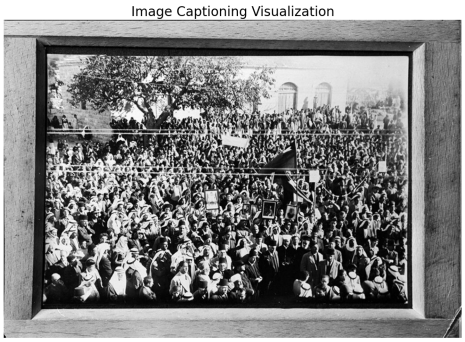


Figure 4: A photo of a historical demonstration in support of Palestine.

and the training camp. However, it also introduces factual errors by naming two additional individuals (Farouk Qaddoumi and Ismail Shammout) who are not confirmed to be present in the image. The AraViT-GPT model captioned the image as: **"An image of a parade of Palestinian Liberation Army soldiers in one of the training camps"**, which is more generic, omits any individual identification, and focuses solely on the setting.

Figure 5: A photo of a training camp involving members of the Palestinian revolution and Yasser Arafat in the middle of the group

6 Ablation study

As part of our ablation study, we implemented an intermediate image captioning system using a Vision-Encoder–Decoder architecture. This was not the final model reported in our main results, but it served to evaluate the performance trade-offs of combining a vision transformer encoder with a transformer-based autoregressive decoder.

Encoder The encoder component uses the `google/vit-base-patch16-224` Vision Transformer (Dosovitskiy et al., 2021), which processes input images into a fixed-length sequence of visual embeddings. Images are preprocessed using `ViTImageProcessor` to ensure consistent scaling, normalization, and patch segmentation.

Decoder The decoder is initialized from a pretrained GPT-2 model (Radford et al., 2019). Since GPT-2 was originally trained with an English tokenizer, we replace its tokenizer with `aubmindlab/bert-base-arabertv2` (Antoun et al., 2020) to enable high-quality Arabic caption generation. The decoder's embedding layer is resized to match the Arabic tokenizer's vocabulary size, and a new padding token is introduced to handle sequence batching.

Tokenizer Adaptation To accommodate GPT-2's architecture (Radford et al., 2019) with the Arabic tokenizer, the model configuration is updated to set `decoder_start_token_id`, `eos_token_id`, and `pad_token_id` appropriately. This ensures proper autoregressive decoding in Arabic.

Integration The encoder's output embeddings are passed to the decoder through the `VisionEncoderDecoderModel` framework from HuggingFace Transformers (Wolf et al., 2020). Training optimizes the cross-entropy loss over token predictions, ignoring padding tokens via masking.

7 Conclusion

In this paper, we developed CRAFT, a custom Arabic image captioning model that integrates Faster R-CNN region features, a custom vision transformer encoder, and a transformer decoder. The system was trained on a custom human-annotated dataset focused on the Palestinian narrative. We also developed AraViT-GPT, another Arabic captioning model, to evaluate against CRAFT. The evaluation results show that CRAFT excelled in semantic similarity, which was our primary metric, achieving a cosine similarity score of 56.22 on the test set. In contrast, AraViT-GPT achieved slightly higher BLEU and LLM judge scores.

We also demonstrated that CRAFT was able to identify people, locations, artifacts, and many other objects in the images. Despite the strengths of CRAFT, it has limitations, including occasional factual inaccuracies in named entities and limited lexical overlap with reference captions.

Future work will focus on scaling the dataset with more diverse Arabic captions, refining named entity recognition through multimodal pretraining, and incorporating nucleus sampling to improve caption fluency.

8 Limitations

While our approach achieved a good performance, several limitations remain. First, the dataset size is relatively small compared to standard benchmarks in image captioning, which restricts the generalization capacity of large-scale transformer models. Second, the captions are single reference annotations; it would be better for the model to have multiple references per image in order to capture the variability of natural language and allow fairer evaluation. Finally, our experiments were conducted on a single GPU (NVIDIA T4), which constrained the scale of hyperparameter exploration and limited the feasibility of training larger models.

9 Acknowledgments

We acknowledge the organizers of the shared task, the many contributors and annotators who enabled this work, and the anonymous reviewers whose comments improved the manuscript.

References

Moustafa Al-Hajj and Mustafa Jarrar. 2022. Arabglossbert: Fine-tuning bert on context-gloss pairs for wsd. *arXiv preprint arXiv:2205.09685*.

Rabee Al-Qasem, Mohannad Hendi, and Banan Tantour. 2025. Alkafi-llama: Fine-tuning llms for precise legal understanding in palestine. *Discover Artificial Intelligence*, 5(107).

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. Arabert: Transformer-based model for arabic language understanding. In *LREC*.

Ahlam Bashiti, Alaa Aljabari, Hadi Hamoud, Md. Rafiul Biswas, Bilal Shalash, Mustafa Jarrar, Fadi Zaraket, George Mikros, Ehsaneddin Asgari, and Wajdi Zaghouani. 2025. ImageEval 2025: The First Arabic Image Captioning Shared Task. In *Proceedings of the Third Arabic Natural Language Processing Conference (ArabicNLP 2025)*, Suzhou, China. Association for Computational Linguistics.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, and et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Mohannad Mohammad Izzat Hendi and 1 others. 2023. *Accurate Pedestrian Detection for Human Crowds Using Deep Learning Techniques*. Ph.D. thesis, AAUP.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Sébastien Marcel and Yann Rodriguez. 2010. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Thomas Wolf, Lysandre Debut, Victor Sanh, and et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

A Appendix

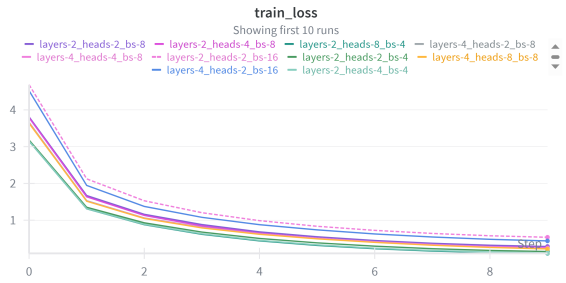
A.1 Final parameters used in training

Table 3: Training hyperparameters used in the experiments.

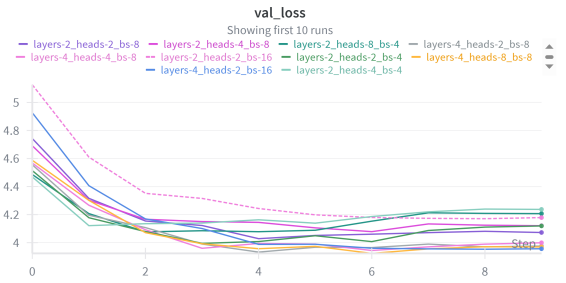
Parameter	Value
Number of epochs	40
Batch size	8
Encoder layers	2
Decoder layers	2
Attention heads (encoder)	2
Attention heads (decoder)	2
Learning rate	1×10^{-4}
Optimizer	AdamW
Learning rate schedule	Linear
Loss function	Cross Entropy Loss
Max sequence length	97
Input features	50 regions

A.2 Grid search tuning results

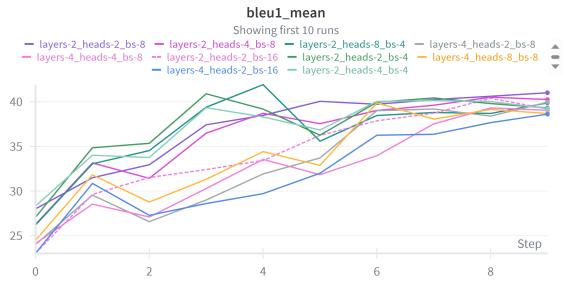
This section presents the results of our grid search experiments, showing how different hyperparameter configurations affected training and validation loss, as well as BLEU, ROUGE, and cosine similarity scores. The plots illustrate the trade-offs that guided our choice of the final configuration.



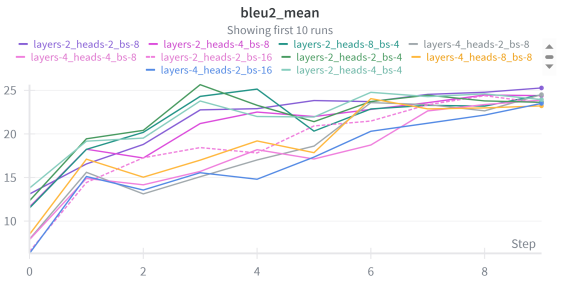
(a) Training loss across hyperparameter configurations.



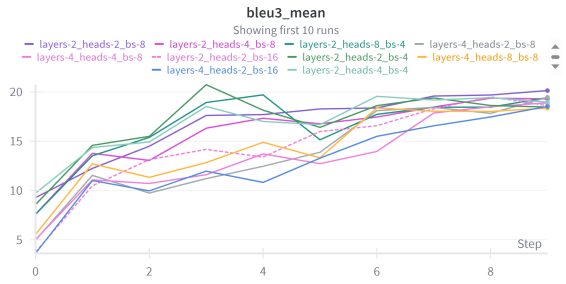
(b) Validation loss across hyperparameter configurations.



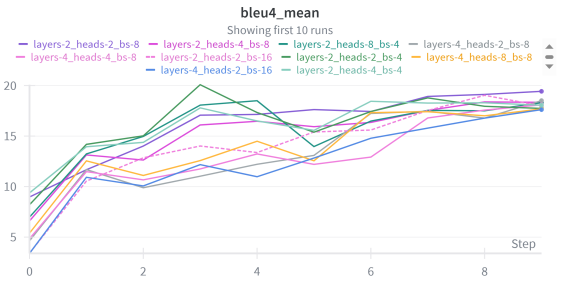
(c) BLEU-1 scores.



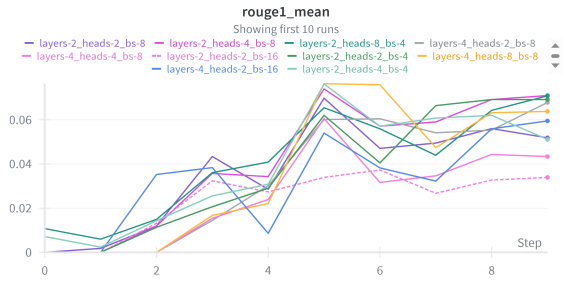
(d) BLEU-2 scores.



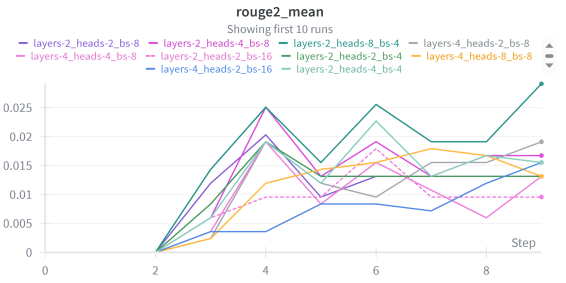
(e) BLEU-3 scores.



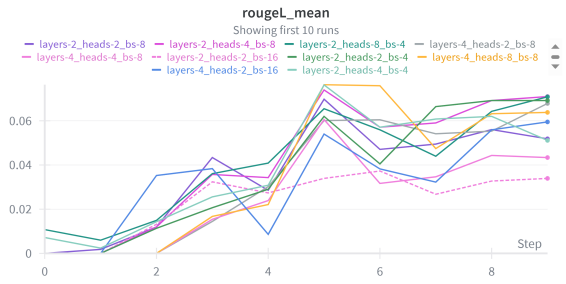
(f) BLEU-4 scores.



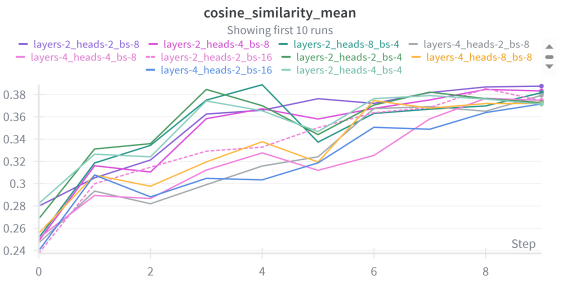
(g) ROUGE-1 scores.



(h) ROUGE-2 scores.



(i) ROUGE-L scores.



(j) Cosine similarity across configurations.

Figure 6: Loss curves and evaluation metrics across hyperparameter configurations during the grid search.