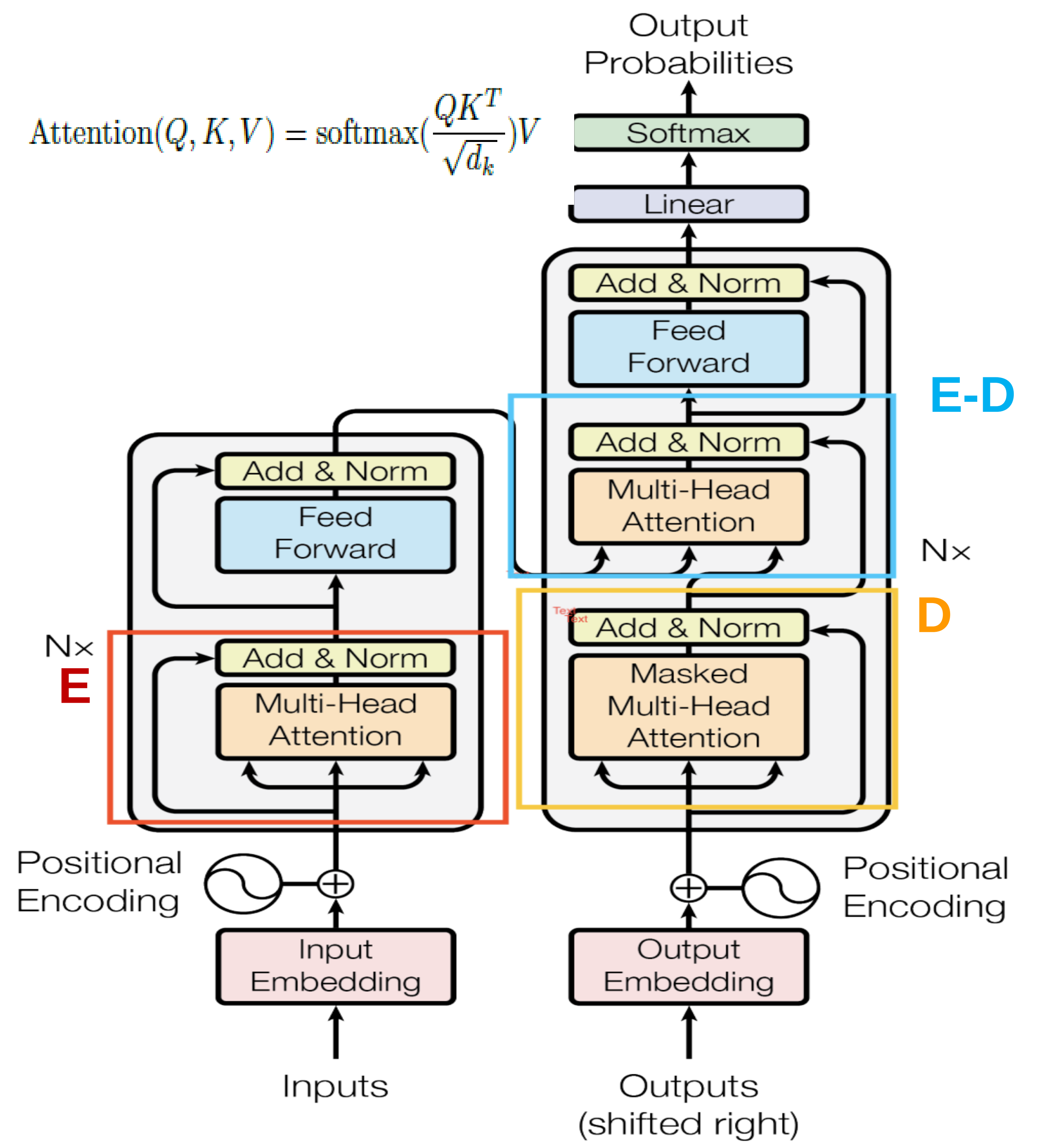
Attentions for Long Document Summarization

Sai Sahana Bhargavi Byrapu Mrudula Adira Bantwal Shreyas Mallya Medha Bharadwaj

Background

NLP models are used to generate summaries and this task becomes difficult when handling long lengths of data that might lead to (1) loss of context (2) Bad summary quality and (3) Model inefficiency.
Attention mechanisms add some context to words in the sentence and use pairwise words relations between tokens and this can be used to improve the summarization models. Frameworks like Pre-trained seq2seq Transformers take O(n^2) time complexity. To reduce such quadratic time in **self-attention ,** selectively attending to neighboring tokens (or) relevant words helps. Yet, these methods do not apply to **encoder-decoder attentions** in summarization models since they collaborate and dynamically pinpoint salient content in the source as the summary is decoded.
Truncation is commonly used to circumvent the issue. However, training on curtailed content further aggravates “hallucination” in existing abstractive models. HEPOS helps address this issue.

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

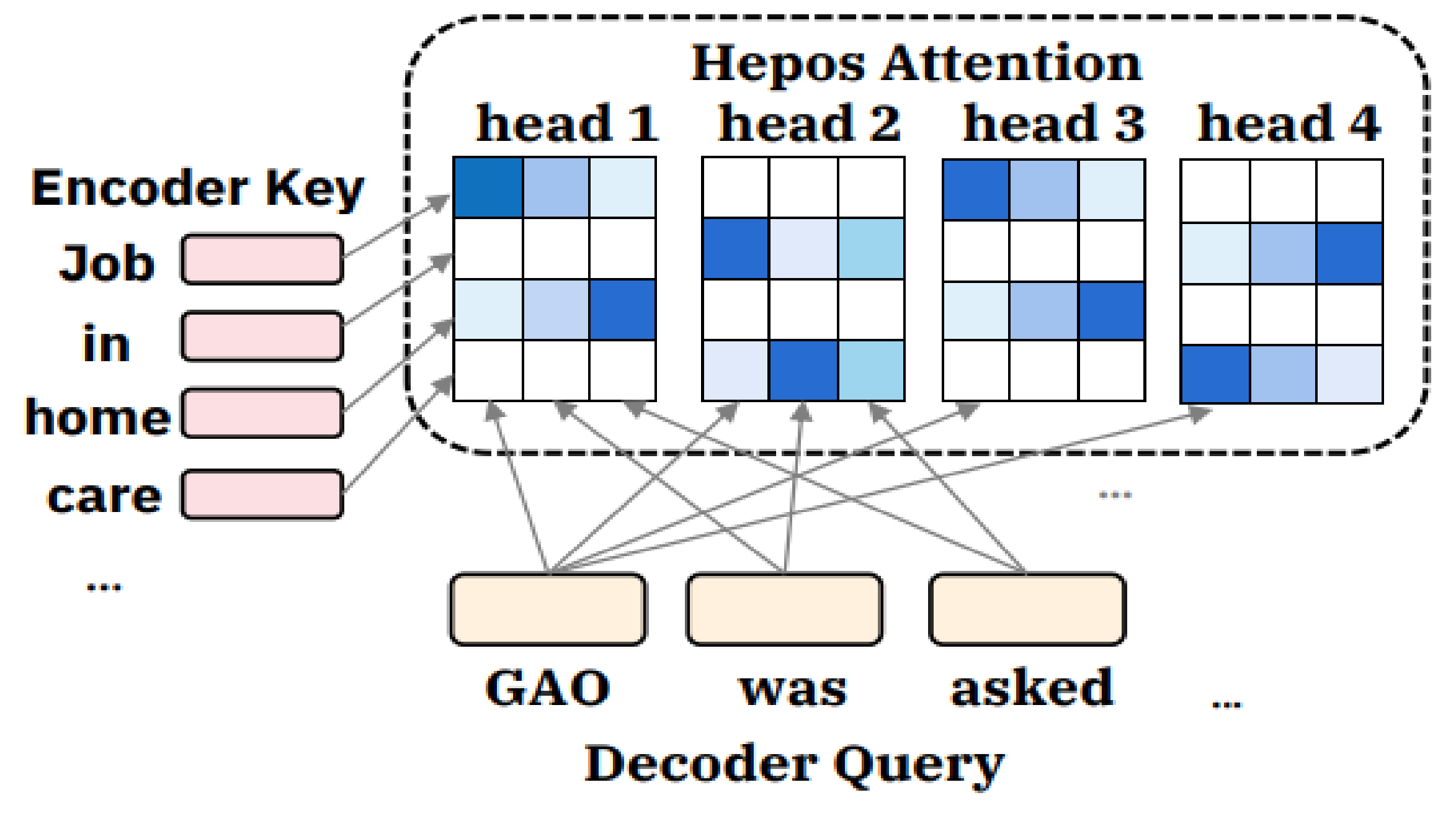


Research Question

The motivation here is to use the HEPOS attention mechanism and ensemble the two best performing summarization models to evaluate if the performance of the ensemble can be better than the individual models.

Approach

Using HEPOS with BART for summarization.
HEPOS: Head-wise Positional Strides
1.To reduce the redundancy of multi-attention heads, HEPOS Uses separate encoder-decoder heads to cover different subsets of source tokens at fixed intervals.
2.Each head starts at a different position, and all heads collectively attend to the full sequence.



Given a stride size of sh, for the h-th head, its attention value between decoder query qj (at step j) and encoder key vector ki (for the i-th input token) can be formulated as:

$$a_{ji}^{h} = \begin{cases} \text{softmax}(\mathbf{q}_j \mathbf{k}_i), & \text{if } (i-h) \bmod s_h = 0 \\ 0 & \text{otherwise} \end{cases}$$

where, each query token attends to n/sh tokens per head and full attention combined with HEPOS is able to process ten times more tokens than existing models that use full attentions.
BART: Bidirectional Auto Regressive Transformer is a denoising autoencoder for pretraining sequence-to-sequence models with bidirectional encoder (like BERT) and a left-to-right decoder (like GPT). rained by (1) corrupting text with an arbitrary noising function, and (2) learning a model to reconstruct the original text.

Dataset

GOVREPORT Dataset containing 19, 466 long reports published by U.S. Government Accountability Office (GAO).
(1) It contains significantly longer documents (9.4k words) and summaries (553 words)
(2) Salient content is spread throughout the documents, as opposed to cases where summary-worthy words are more heavily concentrated in specific parts of the document.
Dataset is split into train, validation and test set by publication date with 17519 training samples, 974 validation documents, and 973 test samples.

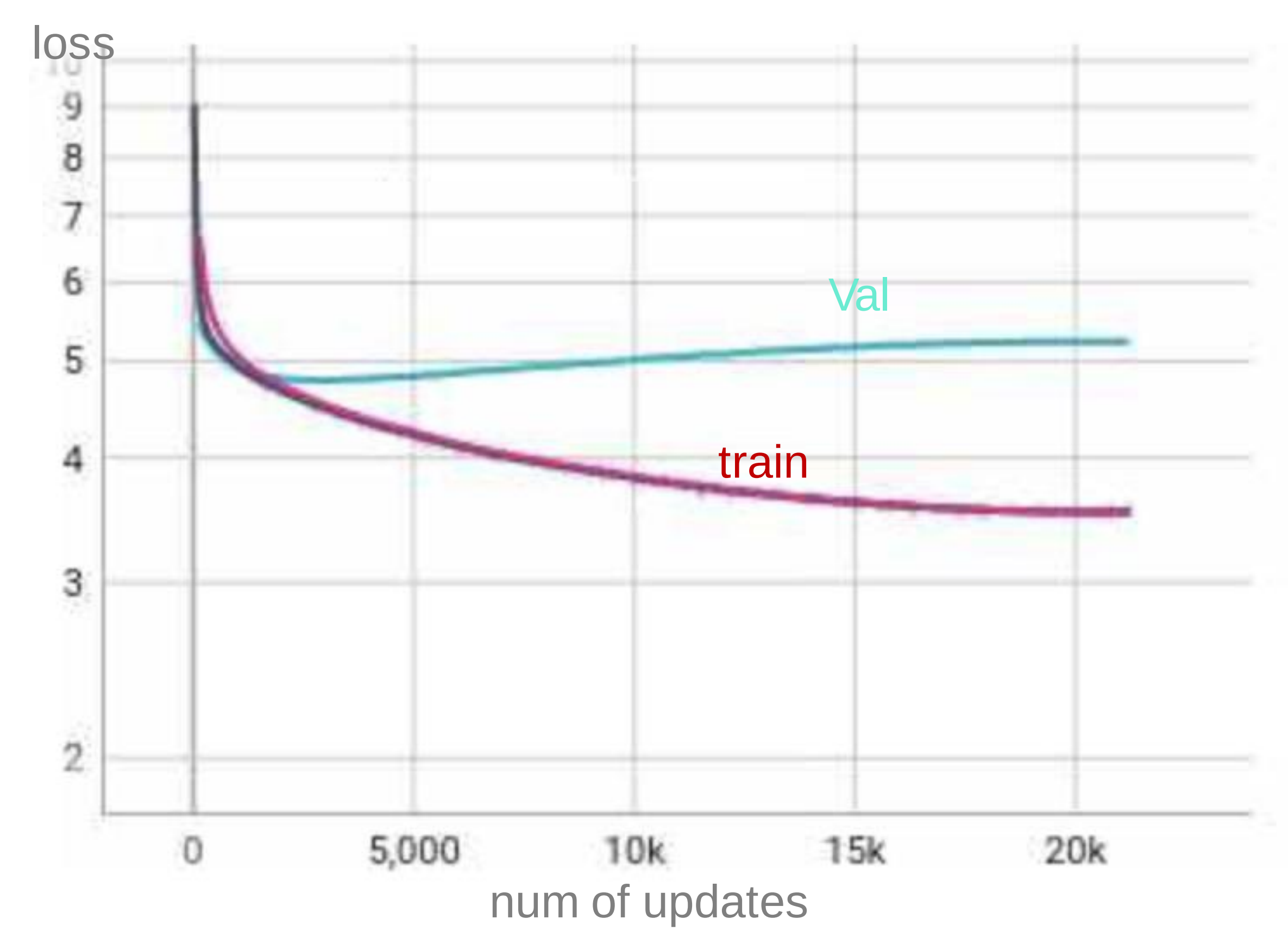
Results

MODEL	ROUGE-1	ROUGE-2	ROUGE-L
BART	43.11	16.92	26.10
HEPOS + BART	51.05	19.44	48.51
RoBERTA	13.77	0.02	12.83
PEGASUS	34.78	4.40	25.00

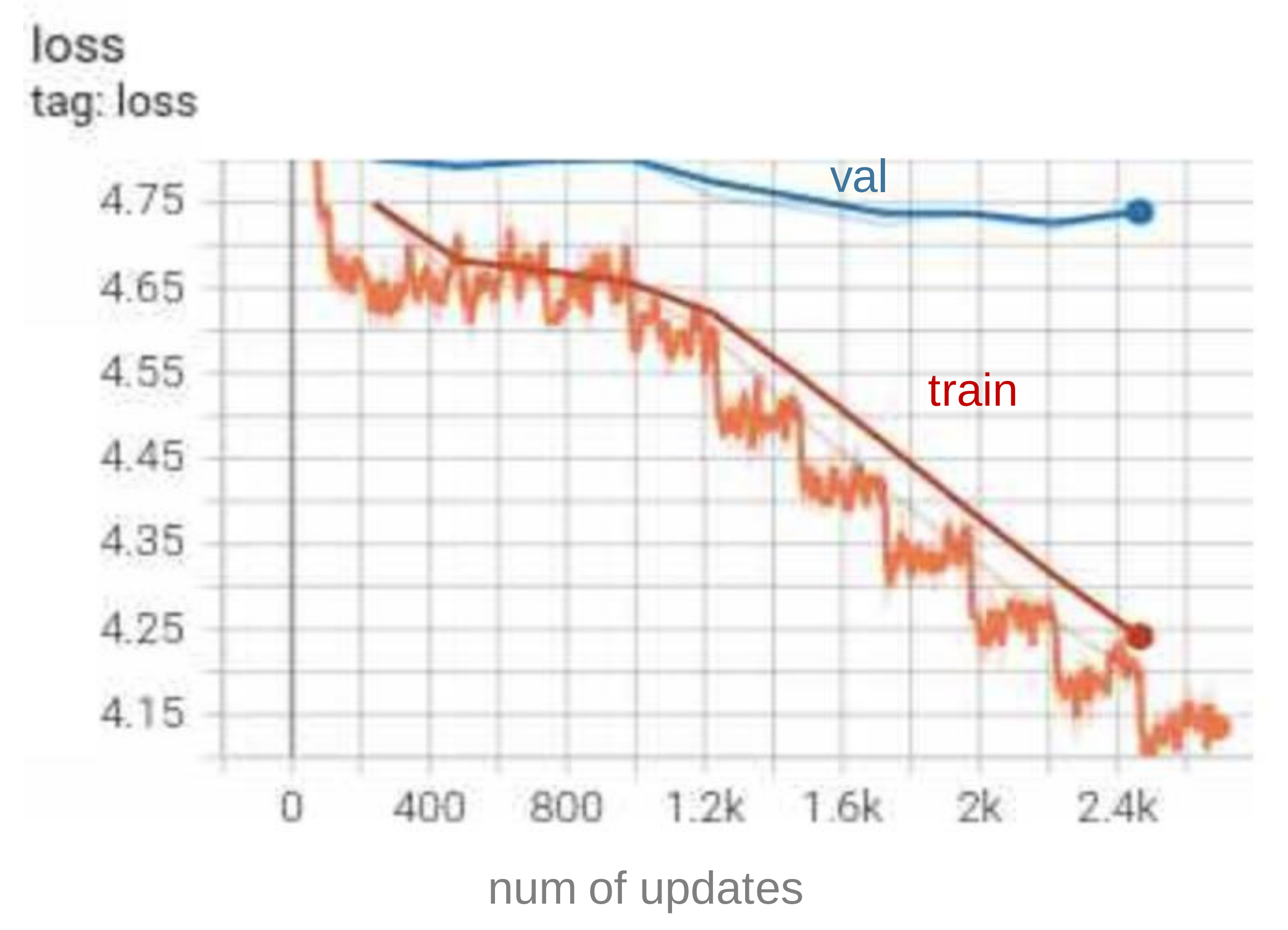
RoBERTA: Is an extension of BERT with certain modifications to the pretraining procedure. It incorporates longer training time over longer sequences, bigger batch size over more data and supports dynamically changing the masking pattern applied on the training data.

PEGASUS: Is a sequence-to-sequence model with gap-sentences generation as a pretraining objective, tailored for abstractive text summarization. In this technique important sentences are masked/removed from an input document and are generated as one output sequence from the remaining sentences.

Finetuned BART



Finetuned HEPOS with BART



The learning rate is set to 1 × 10−4 and learning rate warm-up is applied for the first 10,000 steps. Ada-factor optimizer with a gradient clipping of 0.1 is used. Models are trained on 2 X A100 GPU with 400 GB memory. batch size is of 2 per step and accumulated gradient every 32 steps.