Jump to Conclusions: Short-Cutting Transformers with Linear Transformations

Alexander Yom Din[1], **Taelin Karidi**[1], **Leshem Choshen**[1], **Mor Geva**[2]
[1]Hebrew University of Jerusalem [2]Tel Aviv University
{alexander.yomdin, taelin.karidi, leshem.choshen}@mail.huji.ac.il, morgeva@tauex.tau.ac.il

Abstract

Transformer-based language models create hidden representations of their inputs at every layer, but only use final-layer representations for prediction. This obscures the internal decision-making process of the model and the utility of its intermediate representations. One way to elucidate this is to cast the hidden representations as final representations, bypassing the transformer computation in-between. In this work, we suggest a simple method for such casting, using linear transformations. This approximation far exceeds the prevailing practice of inspecting hidden representations from all layers, in the space of the final layer. Moreover, in the context of language modeling, our method produces more accurate predictions from hidden layers, across various model scales, architectures, and data distributions. This allows "peeking" into intermediate representations, showing that GPT-2 and BERT often predict the final output already in early layers. We then demonstrate the practicality of our method to recent early exit strategies, showing that when aiming, for example, at retention of 95% accuracy, our approach saves additional 7.9% layers for GPT-2 and 5.4% layers for BERT. Last, we extend our method to linearly approximate sub-modules, finding that attention is most tolerant to this change. Our code and learned mappings are publicly available at https://github.com/sashayd/mat.

Keywords: interpretability, language models, efficiency, logitlens, linear lense, linear, early exit, shortcut, layer jump

1. Introduction

Transformer-based language models (LMs) process an input sequence of tokens by first representing it as a sequence of vectors and then repeatedly transforming it through a fixed number of attention and feed-forward network (FFN) layers (Vaswani et al., 2017). While each transformation creates new representations, only the final representations are used to obtain model predictions. Correspondingly, LM loss minimization directly optimizes the final representations, while hidden representations are only optimized implicitly, thus making their interpretation and usefulness more obscure.

However, utilizing hidden representations is highly desirable; a successful interpretation of them can shed light on the "decision-making process" in the course of transformer inference (Tenney et al., 2019; Voita et al., 2019; Slobodkin et al., 2021; Geva et al., 2022b), and obtaining predictions from them can substantially reduce computational cost (Schwartz et al., 2020; Xu et al., 2021).

Previous attempts to harness hidden representations viewed the hidden representations of an input token as a sequence of approximations of its final representation (Elhage et al., 2021; Geva et al., 2022b). This view is motivated by the additive updates induced via the residual connections (He et al., 2016) around each layer in the network. Indeed, previous works (Geva et al., 2021, 2022a; Ram et al., 2022; Alammar, 2021) followed a simplifying assumption that representations *at any layer* can be transformed into a distribution over the output vocabulary via projection to the output embeddings. While this approach has proven to be surprisingly effective for interpretability (Geva et al., 2022a; Dar et al., 2022) and computation efficiency (Schuster et al., 2022; Xin et al., 2020; Schwartz et al., 2020), it oversimplifies the model's computation and assumes that all the layers in the network operate in the same space.

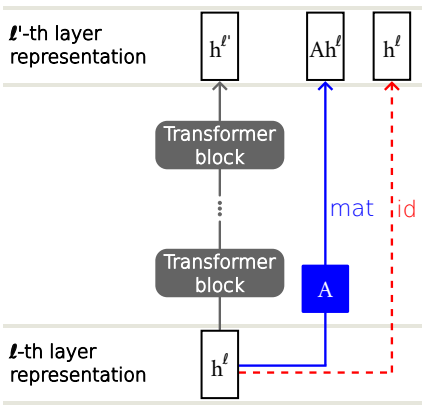

Figure 1: An illustration of our approach to enhance interpretability and utilization of hidden representations. We use linear mappings $A = A_{\ell',\ell}$ to short-cut transformer inference in-between layers ($\texttt{mat}_{\ell\to\ell'}$), instead of the prevalent baseline of propagating the hidden representation as-is to the further layer ($\texttt{id}_{\ell\to\ell'}$).

A natural question that arises is whether there is a more accurate way to cast hidden representations

LREC-COLING 2024, pages 9615–9625

into final representation substitutes than interpreting them as they are. In this work, we tackle this question by learning linear transformations across layers in the network (illustrated in Fig. 1). For any two layers $\ell < \ell'$, we fit a linear regression to transform hidden representations from layer ℓ to layer ℓ'. We show that this method, denoted as `mat`, produces substantially more accurate approximations than the above-discussed identity mapping, dubbed `id`, applied in previous works (§3). As `mat` is a non-contextual mapping that operates on single hidden representations, this suggests that there is more linearity to transformer inference than could be estimated by the `id` mapping.

Next, we test if these gains in approximating future representations also translate to better prediction estimations (§4). To this end, we measure how often language modeling predictions from final representation substitutes produced by `mat`, and by alternations between `mat` and regular inference, agree with those of actual final representations. Through experiments with two data sources and various scales of GPT-2 (Radford et al., 2019) and BERT (Devlin et al., 2019), we observe large accuracy gains (15%-40% at most layers) in prediction estimation by `mat` over naive projections (`id`). Moreover, we show that our mappings generalize well across different data distributions (§5).

We leverage these findings for enhancing model efficiency and demonstrate our method's utility in the setting of early exiting – a strategy for dynamically deciding at which layer to stop the inference pass and use that layer's representation for prediction. While previous works have utilized these hidden representations intact (i.e. using `id`), we transform them using `mat`, showing that our method performs better than the baseline in this setting as well (§6), allowing for the saving of additional 7.9% (resp. 5.4%) of the layers for GPT-2 (resp. BERT) when aiming at 95% accuracy.

Last, we analyze how well the different sub-modules of transformer computation – attention, FFN, and layer normalization – can be estimated linearly (§7), by applying the same methodology of linear mappings. We find that linearly approximating attention, the only sub-module that has contextual processing, results in the least reduction of precision. This hints at an interesting possibility of compute time reduction, as non-contextual inference is parallelizable.

To conclude, we propose a method for casting hidden representations across transformer layers, that is light to train, cheap to infer, and provides more accurate and robust representation approximations than the commonly-used baseline of identical propagation. Beyond interpretability, our method holds potential for enhancing efficiency.

2. Background and Notation

The input to a transformer-based LM (Vaswani et al., 2017) is a sequence of tokens $t_1, ..., t_n$ from a vocabulary $\mathcal{V}$ of size $|\mathcal{V}| = d_v$. The tokens are first represented as vectors using an embedding matrix $E \in \mathbb{R}^{d_h \times d_v}$, where d_h is the hidden dimension of the model, to create the initial *hidden representations*

$$H^0 = (h_1^0, \ldots, h_n^0) \in \mathbb{R}^{d_h \times n}.$$

These representations are then repeatedly transformed through L transformer blocks, where each block outputs hidden representations that are the inputs to the next block:

$$\forall \ell \in [1, L]: \quad \mathtt{b}^\ell(H^{\ell-1}) = H^\ell$$

where

$$H^\ell = (h_1^\ell, \ldots, h_n^\ell) \in \mathbb{R}^{d_h \times n}.$$

The ℓ-th transformer block is constructed as a composition of two layers:

$$\mathtt{b}_\ell = \mathtt{b}_\ell^{\mathtt{ffn}} \circ \mathtt{b}_\ell^{\mathtt{attn}},$$

where $\mathtt{b}_\ell^{\mathtt{attn}}$ (resp. $\mathtt{b}_\ell^{\mathtt{ffn}}$) is a multi-head self-attention (MHSA) layer (resp. FFN layer) enclosed by a residual connection, and potentially interjected with layer normalization (Ba et al., 2016). The *final representations*,

$$H^L = (h_1^L, \ldots, h_n^L),$$

are the transformer stack's output, used to form various predictions. In this work, we investigate whether and how hidden representations from *earlier layers* can be utilized for this purpose instead.

3. Linear Shortcut Across Blocks

To use a hidden representation from layer $\ell < L$ as a final representation, we propose to cast it using linear regression, while skipping the computation in-between these layers. More generally, this approach can be applied to cast any ℓ-th hidden representation to any subsequent layer $\ell' > \ell$.

3.1. Method

Given a source layer ℓ and a target layer ℓ' such that $0 \leq \ell < \ell' \leq L$, our goal is to learn a mapping from hidden representations at layer ℓ to those at layer ℓ'. To this end, we first collect a set of corresponding hidden representation pairs $(h^\ell, h^{\ell'})$. Concretely, we run a set $\mathcal{T}$ of input sequences through the model, and for each input s, we extract the hidden representations $h_{i_s}^\ell, h_{i_s}^{\ell'}$, where i_s is a random position in s. Next, we learn a matrix $A_{\ell',\ell} \in \mathbb{R}^{d_h \times d_h}$

by fitting linear regression over $\mathcal{T}$, i.e., $A_{\ell',\ell}$ is a numerical minimizer for:

$$A \mapsto \sum_{s \in \mathcal{T}} ||A \cdot h^{\ell}_{i_s} - h^{\ell'}_{i_s}||^2,$$

and define the mapping of a representation h from layer ℓ to layer ℓ' as:

$$\texttt{mat}_{\ell \to \ell'}(h) \coloneqq A_{\ell',\ell} \cdot h. \quad (1)$$

3.2. Baseline

We evaluate the prevalent approach of "reading" hidden representations directly, without any transformation. Namely, the propagation of a hidden representation from layer ℓ to layer ℓ' is given by the identity function, dubbed `id`:

$$\texttt{id}_{\ell \to \ell'}(h) \coloneqq h.$$

This baseline assumes that representations at different layers operate in the same linear space.

3.3. Quality of Fit

We first evaluate our method by measuring how well the learned linear mappings approximate the representations at the target layer. To this end, we calculate the (coordinate-averaged) r^2-score of our mapping's outputs with respect to the representations obtained from a full inference pass, and compare to the same for the `id` baseline.

Models. We use GPT-2 (Radford et al., 2019), a decoder-only auto-regressive LM, with $L = 48$, $d_h = 1600$, and BERT (Devlin et al., 2019), an encoder-only model trained with masked language modeling, with $L = 24$, $d_h = 1024$.

Data. We sample random sentences from Wikipedia, collecting 9,000 (resp. 3,000) sentences for the training set $\mathcal{T}$ (resp. validation set $\mathcal{V}$).[1] For each example s, we select a random position i_s and extract the hidden representations $h^{\ell}_{i_s}$ at that position from all the layers. For BERT, we first replace the input token at position i_s with a `[MASK]` token, as our motivation is interpreting predictions, which are obtained via masked tokens in BERT (see §4.2). Thus, in this case, the hidden representations we consider are of `[MASK]` tokens only.

Evaluation. For every pair of layers ℓ, ℓ', such that $0 \leq \ell < \ell' \leq L$, we use the training set $\mathcal{T}$ to fit linear regression as described in §3.1, and obtain a mapping $\texttt{mat}_{\ell \to \ell'}$. Next, we evaluate the quality of $\texttt{mat}_{\ell \to \ell'}$ as well as of $\texttt{id}_{\ell \to \ell'}$ using the r^2-coefficient, uniformly averaged over all coordinates. Concretely, we compute the r^2-coefficient of each of the predicted representations $\texttt{mat}_{\ell \to \ell'}(h^{\ell}_{i_s})$ and $\texttt{id}_{\ell \to \ell'}(h^{\ell}_{i_s})$ versus the true representations $h^{\ell'}_{i_s}$ over all $s \in \mathcal{V}$.

[1] We use sentences rather than full documents to simplify the analysis.

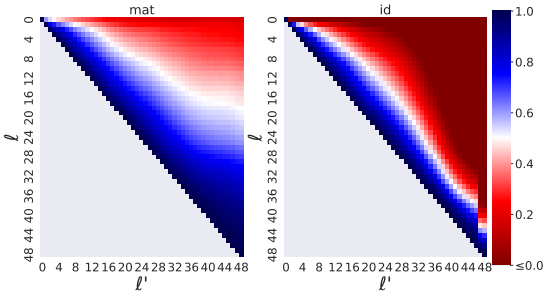

Figure 2: The coordinate-averaged r^2-score of $\texttt{mat}_{\ell \to \ell'}$ (left) and $\texttt{id}_{\ell \to \ell'}$ (right) (GPT-2).

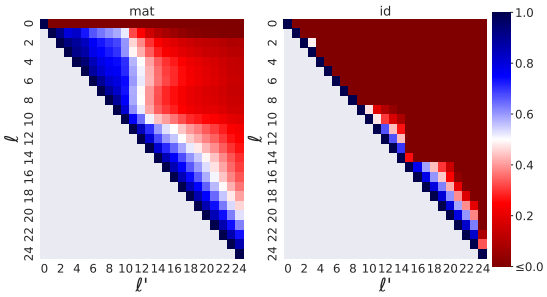

Figure 3: The coordinate-averaged r^2-score of $\texttt{mat}_{\ell \to \ell'}$ (left) and $\texttt{id}_{\ell \to \ell'}$ (right) (BERT).

Results. Results for GPT-2 and BERT are presented in Figs. 2 and 3, respectively. In both models, `mat` consistently yields better approximations than `id`, as it obtains higher r^2-scores (in blue) across the network. This gap between `mat` and `id` is especially evident in BERT, where `id` completely fails to map the representations between most layers, suggesting that hidden representations are modified substantially by every transformer block. Overall, this highlights the shortcoming of existing practices to inspect representations in the same linear space, and the gains from using our method to approximate future layers.

4. Linear Shortcut for Language Modeling

We saw that our method approximates future hidden representations substantially better than a naive propagation. In this section, we show that this improvement also translates to better predictive abilities from earlier layers. Concretely, we use our method to estimate the final prediction from intermediate representations, in the context of two fundamental LM tasks; next token prediction and masked token prediction.

Evaluation Metrics. Let $h, h' \in \mathbb{R}^{d_h}$ be a final representation and its substitute obtained by some mapping, and denote by $\delta(h), \delta(h') \in \mathbb{R}^{d_v}$ their corresponding output probability distributions (see details below). We measure the prediction quality of h' with respect to h using two metrics:

- **Precision@k** (↑ is better): This checks whether the token with the highest probability according to $\delta(h')$ appears in the top-k tokens according to $\delta(h)$. Namely, we sort $\delta(h)$ and assign a score of 1 if $\arg\max(\delta(h'))$ appears in the top-k tokens by $\delta(h)$, and 0 otherwise.
- **Surprisal** (↓ is better): We measure the negative log likelihood according to $\delta(h)$, of the highest-probability token according to $\delta(h')$. Intuitively, low values mean that the model sees the substitute result as probable and hence not surprising.

We report the average Precision@k and Surprisal over the validation set $\mathcal{V}$.

4.1. Next Token Prediction

Auto-regressive LMs output for every position a probability distribution over the vocabulary for the next token. Specifically, the output distribution for every position i is given by $\delta(h_i^L)$, where

$$\delta(h) = \texttt{softmax}(E^\top \cdot h) \in \mathbb{R}^{d_v}. \quad (2)$$

For some LMs, including GPT-2, a layer normalization `ln_f` is applied to the final layer representation before this conversion (i.e., computing $\delta(\texttt{ln\_f}(h))$ rather than $\delta(h)$).

Recall that our goal is to measure how well this distribution can be estimated from intermediate representations, i.e. estimating $\delta(h_i^L)$ from h_i^ℓ where $\ell < L$. Thus, we first run the validation set examples through the model, while extracting for each example s and every layer the hidden representation at a random position i_s. Next, we apply our mappings $\texttt{mat}_{\ell\to L}$ and $\texttt{id}_{\ell\to L}$ to cast the hidden representations of every layer ℓ to final layer substitutes (see §3). Last, we convert every final-layer substitute to an output distribution (Eq. 2) and compute for each layer the average Precision@k (for $k = 1, 5, 10$) and Surprisal scores with respect to the final output distribution, over the validation set.

Results. Fig. 4 shows the average Precision@k and Surprisal scores per layer in GPT-2. Across all layers, `mat` outperforms `id` in terms of both scores, often by a large margin (e.g. till layer 44 the Precision@1 achieved by `mat` is bigger than that of `id` by more than 20%). This shows that linear mappings enable not just better estimation of final layer representations, but also of the predictions they induce. Moreover, the relatively high Precision@k scores of `mat` in early layers (62%-82% for $k = 10$, 52%-74% for $k = 5$, and 28%-45% for $k = 1$) suggest that early representations often accurately approximate the final prediction. Also, the substantially lower Surprisal scores of `mat` compared to `id` imply that our method allows for a more representative reading into the layer-wise prediction-formation of the model than allowed via direct projection to the vocabulary.

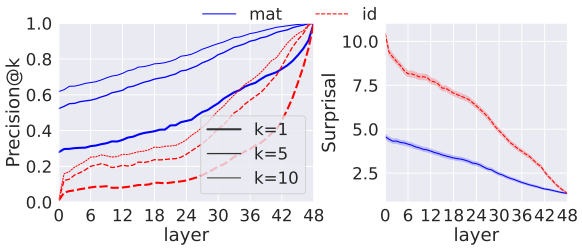

Figure 4: Precision@k ($k = 1, 5, 10$) and Surprisal for $\texttt{mat}_{\ell\to L}$ and $\texttt{id}_{\ell\to L}$ (GPT-2 next token prediction task). 95% confidence intervals are shown for Surprisal.

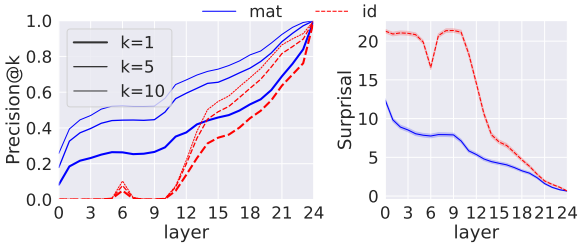

Figure 5: Precision@k ($k = 1, 5, 10$) and Surprisal for $\texttt{mat}_{\ell\to L}$ and $\texttt{id}_{\ell\to L}$ (BERT masked token prediction task). 95% confidence intervals are shown for Surprisal.

4.2. Masked Token Prediction

We conduct the same experiment in §4.1 for masked language modeling, where the model predicts a probability distribution for a masked token in the input. Unlike next token prediction, where the output distribution is computed from representations of varying input tokens, in masked token prediction the output is always obtained from representations of the same input token (i.e. `[MASK]`).

For this experiment, we use BERT, on top of which we use a pretrained masked language model head δ; given a token sequence s, a `[MASK]` token inside it and its final representation h, $\delta(h) \in \mathbb{R}^{d_v}$ is a probability distribution over tokens giving the model's assessment of the likelihood of tokens to be fitting in place of the `[MASK]` token in s.

Results. Fig. 5 shows the average Precision@k and Surprisal scores per layer in BERT, overall showing trends similar to those observed for next token prediction in GPT-2 (§4.1). This is despite the

$\texttt{id}_4$	$\texttt{mat}_4$	$\texttt{id}_{12}$	$\texttt{mat}_{12}$	$\texttt{id}_{24}$
Input: *aldridge had shoulder surgery in* `[MASK]`.				
fellowship	time	cyclist	2009	september
employment	it	emergencies	2008	november
agreement	her	seniors	2010	december
##ostal	them	cycling	2006	august
##com	work	pennsylvania	2007	july
Input: *on your next view you will be asked to* `[MASK]` *continue reading.*				
##com	be	be	be	please
accreditation	get	undergo	please	simply
©	go	spartans	help	also
fellowship	help	seniors	simply	again
summer	have	*	say	immediately

Table 1: Examples of top-5 BERT masked token predictions at layers 4, 12 and 24, under the mappings $\texttt{mat}_{\ell\to L}$ (abbreviated $\texttt{mat}_\ell$) and $\texttt{id}_{\ell\to L}$ (abbreviated $\texttt{id}_\ell$). Plausible predictions (according to a human annotator) are marked in blue. Note that for $\ell = L = 24$, predictions of $\texttt{mat}_\ell$ and $\texttt{id}_\ell$ are the same.

differences between the two tasks and the considerable architectural differences between the models. Notably, the superiority of `mat` over `id` in this setting is even more prominent; while, in the first ten layers, `mat`'s precision is between 8%-52% (Fig. 5), `id`'s precision for all values of k is close to zero, again strongly indicating that our method allows for better reading into early layer hidden representations. More generally, `mat` improves the Precision@1 of `id` by more than 17% at most layers, and unveils that a substantial amount of predictions ($> 25\%$ starting from layer 3) appear already in the very first layers. Interestingly, the (rough) divide between the first last halves of layers for `id` in Fig. 5 seems to align with the two-hump shape of the blue region for `mat` in Fig. 3.

Analysis. We manually compare the predictions of our mapping $\texttt{mat}_{\ell\to L}$ with $\texttt{id}_{\ell\to L}$, for the BERT model. Concretely, we select 50 random sentences from the Leipzig dataset (see §5.2). Next, for each layer ℓ, we manually analyze how many of the top-5 tokens according to $\texttt{mat}_{\ell\to L}$ and $\texttt{id}_{\ell\to L}$ fit into context. We consider a token to fit into context if it is grammatically plausible within the sentence (see Tab. 1 for examples). In the resulting 1,250 instances (i.e. 50 sentences $\times$ 25 representations), we observe a substantially higher plausibility rate of 85.4% for `mat` compared to 52.8% for `id`. In fact, only in less than 4.3% of the instances, there are more plausible tokens among the top-5 tokens according to `id` than according to `mat`, further supporting the Surprisal results above.

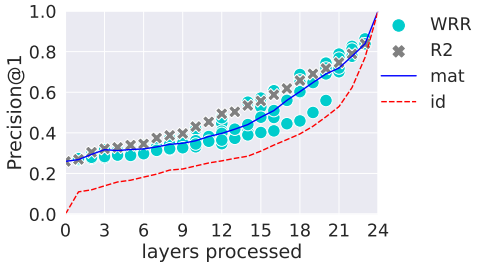

Figure 6: Precision@1 for various alternation schemes and previous mappings for comparison (24-layer GPT-2 next token prediction task).

4.3. Alternation Schemes

Thus far, we considered direct mappings to the last layer. We now check if mappings between intermediate layers can improve prediction estimations further. To this end, we obtain final-representation substitutes by alternating between transformer-inference and linear mappings. For $\ell < \ell'$, let us abbreviate

$$\texttt{b}_{\ell\to\ell'} := \texttt{b}^{\ell'} \circ \cdots \circ \texttt{b}^{\ell+2} \circ \texttt{b}^{\ell+1},$$

i.e. $\texttt{b}_{\ell\to\ell'}$ is the application of transformer inference from layer ℓ to layer ℓ'. For a sequence $0 = \ell_0 < \ell_1 < \ldots < \ell_n = L$, we consider either of

$$\cdots \circ \texttt{b}_{\ell_2\to\ell_3} \circ \texttt{mat}_{\ell_1\to\ell_2} \circ \texttt{b}_{\ell_0\to\ell_1}, \quad (3)$$

$$\cdots \circ \texttt{mat}_{\ell_2\to\ell_3} \circ \texttt{b}_{\ell_1\to\ell_2} \circ \texttt{mat}_{\ell_0\to\ell_1}. \quad (4)$$

In other words, those are inference modes alternating between transformer inference and application of our linear mappings in a prescribed manner. We then collect two sets of alternation schemes:

- **r^2-informed (R2)**: We define the r^2-score of Eq. 3 (resp. Eq. 4) to be the product of the r^2-scores of $\texttt{mat}_{\ell_i\to\ell_{i+1}}$ (computed in §3.3), for $i = 1, 3, \ldots$ (resp. $i = 0, 2, \ldots$). For each ℓ, we consider the scheme that employs ℓ transformer blocks with the maximal r^2-score.
- **Weighted round-robin (WRR)**: For $a, b \geq 1$ such that $a+b$ divides L, we consider $(\ell_0, \ell_1, \ldots)$ given by $(0, a, a+b, 2a+b, 2a+2b, \ldots)$ and the two corresponding schemes Eq. 3, 4. In other words, here we alternate between performing a transformer blocks and application of our linear mapping across b layers, for some fixed values of a and b.

For the experiment, we use GPT-2 with 24 layers (see §5.1) and measure Precision@1.

Results. Fig. 6 presents Precision@1 for various alternation schemes. We see, first of all, that some alternation schemes provide better precision than the previously considered $\texttt{mat}_{\ell\to L}$. Second, the best-r^2-score tactic for choosing an alternation

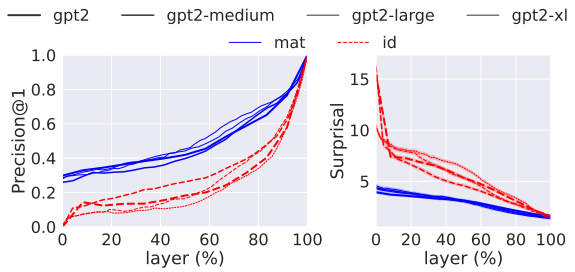

Figure 7: Precision@1 and Surprisal for $\texttt{mat}_{\ell \to L}$ and $\texttt{id}_{\ell \to L}$, for next token prediction with GPT-2. 95% confidence intervals are shown for Surprisal.

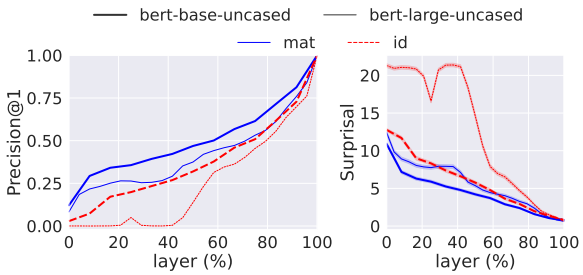

Figure 8: Precision@1 and Surprisal for $\texttt{mat}_{\ell \to L}$ and $\texttt{id}_{\ell \to L}$, for masked token prediction with BERT. 95% confidence intervals are shown for Surprisal.

scheme seems to work well for the first half of layers, but under-achieves (relative to other possible alternation schemes) for the second half. It is, therefore, interesting to try to devise more clever tactics in the future; perhaps, for example, by weighting r^2-scores according to layer index.

5. Method Robustness

5.1. Robustness Across Model Scales

We repeat our experiments in §4, with three additional scales of GPT-2 and one additional scale of BERT. Overall, the models are `gpt2` ($L = 12$, $d_h = 768$), `gpt2-medium` ($L = 24$, $d_h = 1024$), `gpt2-large` ($L = 36$, $d_h = 1280$) and `gpt2-xl` ($L = 48$, $d_h = 1600$), and `bert-base-uncased` ($L = 12$, $d_h = 768$) and `bert-large-uncased` ($L = 24$, $d_h = 1024$).

Fig. 7 (resp. Fig. 8) depicts the Precision@1 and Surprisal scores as functions of the relative depth of the model (i.e. ℓ/L), for GPT-2 models (resp. BERT models). The plots show the same trends observed in §4 across various model scales, with `mat` exhibiting substantially higher predictive abilities from intermediate layers than `id`. Interestingly, there is a great overlap between GPT-2 scores of different scales, but not between the scores of BERT models.

5.2. Robustness Across Data Distributions

We test whether the linear mappings learned from one data distribution are useful for predictions on other data distributions. To this end, we use a second dataset of news article sentences, the 10K English 2020 news sentences corpus from the Leipzig Corpora Collection (Goldhahn et al., 2012), which we randomly divide into a training set $\mathcal{T}$ consisting of 9,000 examples and a validation set $\mathcal{V}$ consisting of 1,000 examples. For our experiments, we use the 24-layer GPT-2 and BERT models. First, we replicate previous experiments on the Leipzig dataset, obtaining results that are extremely similar; for example, the average (across layers) difference between the Precision@1 score of Wikipedia and Leipzig is 0.3% for GPT-2 and -1.4% for BERT. Next, we use Leipzig (resp. Wikipedia) samples to fit linear mappings $\texttt{mat}_{\ell \to L}$ (as described in §3), and then evaluate these mappings in the context of next-token prediction, on samples from Wikipedia (resp. Leipzig) (as in §4). When swapping the original mappings with those trained on the other dataset, we observe a decrease of 0.1% (resp. increase of 1.1%) relative to the original Precision@1 scores for BERT and a decrease of 5.5% (resp. decrease of 8%) relative to the original Precision@1 scores for GPT-2, on average across layers. Overall, this shows that our method generalizes well to out-of-distribution samples. Moreover, our linear mappings capture general, rather than domain specific, features of the model's inference pass.

6. Implication to Early Exiting

The possibility of approximating the final prediction already in the early layers has important implications for efficiency; applying our linear mapping instead of executing transformer blocks of quadratic time complexity, could save a substantial portion of the computation. In this section, we demonstrate this in the context of early exiting.

When using an early exit strategy (Schwartz et al., 2020; Xin et al., 2020; Schuster et al., 2022), one aims at deciding dynamically at which layer to stop the computation and "read" the prediction from the hidden representation of that layer. More precisely, under a confidence measure paradigm, one decides to stop the computation for a position i at layer ℓ based on a confidence criterion, that is derived from casting the hidden representation h_i^ℓ as a final-layer representation and converting it to an output probability distribution. Specifically, following Schuster et al. (2022), a decision to exit is made if the difference between the highest and the second highest probabilities is bigger than

$$0.9 \cdot \lambda + 0.1 \cdot \exp(-4i/N),$$

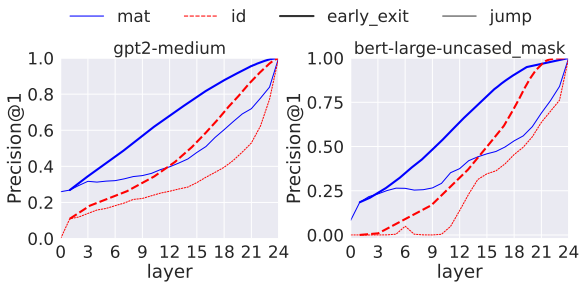

Figure 9: Precision@1 with early exit and "fixed exit", applied to the 24-layer GPT-2 for next token prediction (left) and the 24-layer BERT for masked token prediction (right). Varying the confidence parameter λ, the x-coordinate is the average number of layers processed before an early exit decision is reached.

where N is the average length of the input until position i_s for $s \in \mathcal{V}$, and λ is a hyper-parameter.

Experiment. We assess the utility of our mapping $\texttt{mat}_{\ell \to L}$ for early exit as a plug-and-play replacement for $\texttt{id}_{\ell \to L}$, through which intermediate representations are cast into final-layer representations. We use GPT-2 for the next token prediction and BERT for masked token prediction (both with 24 layers). We run each of the models over the validation set examples, while varying the confidence parameter λ and using either $\texttt{id}_{\ell \to L}$ or $\texttt{mat}_{\ell \to L}$ for casting intermediate representations. Furthermore, we compare these early exit variants to the "fixed exit" strategy from §4, where the computation is stopped after a pre-defined number of layers rather than relying on a dynamic decision. We evaluate each variant in terms of both prediction's accuracy, using the Precision@1 metric (see §4), and efficiency, measured as the average number of transformer layers processed during inference.

Results. Fig. 9 plots the average Precision@1 score against the average number of layers processed, for 24-layer GPT-2 and 24-layer BERT. For both models, under an early exit strategy our mapping $\texttt{mat}$ again provides a substantial improvement over $\texttt{id}$. For example, aiming at 95% average precision, $\texttt{mat}$ saves ~ 3.3 (13.8%) layers in GPT-2 compared to only ~ 1.4 (5.9%) layers by $\texttt{id}$, and ~ 4.8 (20%) layers in BERT versus ~ 3.5 (14.6%) layers by $\texttt{id}$. These results highlight the potential gains prominent early exit methods can obtain by using our method. Notably, in both models and for each of the mapping methods, early exit obtains better results than fixed layer exit, as expected.

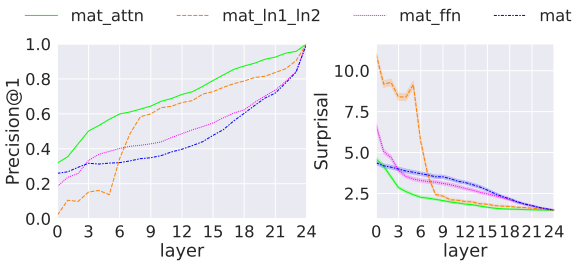

Figure 10: Precision@1 and Surprisal for the various sub-module linear mappings, and $\texttt{mat}_{\ell \to L}$ for comparison (24-layer GPT-2 next token prediction task). A 95% confidence interval surrounds the Surprisal lines.

7. Linear Shortcut Across Sub-Modules

In this section, we investigate whether discrepancies across layers result from specific sub-modules or are a general behaviour of all sub-modules in the network. This is done by extending our approach to test how well particular components in transformer blocks can be linearly approximated.

Method. Consider GPT-2 for definiteness, then:

$$\texttt{b}_\ell = \texttt{b}_\ell^{\texttt{ffn}} \circ \texttt{b}_\ell^{\texttt{attn}}$$

$$\texttt{b}_\ell^{\texttt{attn}}(H) = \texttt{attn}_\ell(\texttt{ln1}_\ell(H)) + H, \qquad (5)$$

where $\texttt{attn}_\ell$ is a MHSA layer and $\texttt{ln1}$ is a layer normalization (LN), and

$$\texttt{b}_\ell^{\texttt{ffn}}(H) = \texttt{ffn}_\ell(\texttt{ln2}_\ell(H)) + H,$$

where $\texttt{ffn}_\ell$ is an FFN layer and $\texttt{ln2}$ is a LN. Given a block $\texttt{b}_\ell$ and one of its sub-modules $\texttt{ln1}_\ell$, $\texttt{attn}_\ell$, $\texttt{ln2}_\ell$, or $\texttt{ffn}_\ell$, we fit linear regression approximating the output of the sub-module given its input, and then use it to define mappings $\texttt{mat\_attn}_{\ell \to \ell'}$, $\texttt{mat\_ln1\_ln2}_{\ell \to \ell'}$ and $\texttt{mat\_ffn}_{\ell \to \ell'}$. We provide the formal definitions of these mappings in App. A.

Evaluation. We analyze the 24-layered GPT-2, and proceed completely analogously to §4.1, evaluating the Precision@1 and Surprisal metrics for the mappings $\texttt{mat\_attn}_{\ell \to L}$, $\texttt{mat\_ffn}_{\ell \to L}$ and $\texttt{mat\_ln1\_ln2}_{\ell \to L}$.

Results. Fig. 10 shows the average Precision@1 and Surprisal scores per layer. From a certain layer (~7), all sub-module mappings achieve better results than the full-block mapping $\texttt{mat}_{\ell \to L}$. Thus, it is not just the cumulative effect of all the sub-modules in the transformer block that is amenable

to linear approximation, but also individual sub-modules can be linearly approximated. Furthermore, the linear approximation of attention sub-modules is less harmful than that of the FFN or LN sub-modules. A possible reason is that the linear replacement of FFN or LN "erodes" the self-attention computation after a few layers. Moreover, the good performance of $\texttt{mat\_attn}_{\ell \to L}$ suggests that contextualization often exhausts itself in early layers; speculatively, it is only in more delicate cases that the self-attention of late layers adds important information. Last, remark the sharp ascent of the scores for layer normalization in layers 5-8, for which we do not currently see a particular reason. To conclude, we see that the possibility of linear approximation permeates transformer components.

8. Related Work

There is a growing interest in utilizing intermediate representations of LMs for interpretability and efficiency. For interpretability, one seeks to understand the prediction construction process of the model (Tenney et al., 2019; Voita et al., 2019; Geva et al., 2022b), or the features stored in its hidden representations (Adi et al., 2017; Conneau et al., 2018; Liu et al., 2019). Our work is different as it converts intermediate representations into a final-layer form, which is interpretable by design.

Previous works on early exiting cut the computation at a dynamically-decided earlier stage (Schwartz et al., 2020; Xin et al., 2020; Schuster et al., 2022; Gera et al., 2023), or a fixed network is utilized to parallelize inference (Leviathan et al., 2022; Chen et al., 2023). However, these methods propagate intermediate representations directly, which we show is substantially worse than our approach. Also, our method requires training of considerably fewer parameters than methods such as Schuster et al. (2021), which learn a different output softmax for each layer.

Last, skipping transformer layers and analyzing the linearity properties of transformer components have been discussed in prior works (Zhao et al., 2021; Mickus et al., 2022; Wang et al., 2022; Lamparth and Reuel, 2023). Specifically, a concurrent work by Belrose et al. (2023) proposed to train affine transformations from hidden to final representations to increase model transparency. Our work is different in that we train *linear* transformations *across all layers*. Moreover, while Belrose et al. (2023) use SGD for training while minimizing KL divergence, we use linear regression, which requires much less compute. It will be valuable to compare the accuracy of both methods.

9. Conclusion and Future Work

We present a simple and effective method for enhancing utilization of hidden representations in transformer-based LMs, that uses pre-fitted context-free and token-uniform linear mappings. Through a series of experiments on different data sources, model architectures and scales, we show that our method consistently outperforms the prevalent practice of interpreting representations in the final-layer space of the model, yielding better approximations of succeeding representations and the predictions they induce, thus allowing a more faithful interpretation of the model's prediction-formation. We demonstrate the practicality of our method for improving computation efficiency, saving a substantial amount of compute on top of prominent early exiting approaches. Also, by extending our method to sub-modules, we observe that replacing a part of the transformer inference by a non-contextual linear computation often results in a small deterioration of the prediction. This opens new research directions for improving model efficiency, including breaking the computation into parallel tasks.

10. Limitations

First, while it is possible to define many different mappings in-between layers, for example, affine or non-linear transformations, we focus on the "simple" case of linear transformations. Our choice is motivated by the wide success of the simplest mapping (i.e. the identity baseline, of inspecting hidden representations in the same linear space), while we are asking if there is more linearity in transformer inference that can be exploited for interpretability.

Second, we find that there is more linear structure to parts of the transformer computation (both full layers and sub-modules) than could be explained solely by the residual connection. However, we do not elucidate a reason for that, leaving exploration of this interesting research question for future work.

Third, our experiments focus on post-hoc interpretability, that is, analyzing a trained model without changing its weights. Future work should also consider analyzing the utility of such linear mappings when those are integrated into the model training.

Last, in our experiments we use only data in English. Nonetheless, given the comprehensiveness of our experiments and the fact that our method does not rely on any language-specific features, we would expect our findings to hold in other languages as well.

11. Bibliographical References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *International Conference on Learning Representations*.

J Alammar. 2021. Ecco: An open source library for the explainability of transformer language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 249–257, Online. Association for Computational Linguistics.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv:1607.06450v1*.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Lev McKinney, Igor Ostrovsky, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *to appear*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv:2302.01318v1*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2022. Analyzing transformers in embedding space. *arXiv:2209.02535v2*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. In *Transformer Circuits Thread*.

Ariel Gera, Roni Friedman, Ofir Arviv, Chulaka Gunasekara, Benjamin Sznajder, Noam Slonim, and Eyal Shnarch. 2023. The benefits of bad advice: Autocontrastive decoding across model layers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10406–10420, Toronto, Canada. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Guy Dar, Paul Roit, Shoval Sadde, Micah Shlain, Bar Tamir, and Yoav Goldberg. 2022a. LM-debugger: An interactive tool for inspection and intervention in transformer-based language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 12–21, Abu Dhabi, UAE. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022b. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the Leipzig corpora collection: From 100 to 200 languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 759–765, Istanbul, Turkey. European Language Resources Association (ELRA).

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Los Alamitos, CA, USA. IEEE Computer Society.

Max Lamparth and Anka Reuel. 2023. Analyzing and editing inner mechanisms of backdoored language models. *arXiv:2302.12461v1*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2022. Fast inference from transformers via speculative decoding. *arXiv:2211.17192v1*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothee Mickus, Denis Paperno, and Mathieu Constant. 2022. How to dissect a Muppet: The structure of transformer embedding spaces. *Transactions of the Association for Computational Linguistics*, 10:981–996.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Ori Ram, Liat Bezalel, Adi Zicher, Yonatan Belinkov, Jonathan Berant, and Amir Globerson. 2022. What are you token about? dense retrieval as distributions over the vocabulary. *arXiv:2212.10380v1*.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems*.

Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. 2021. Consistent accelerated inference via confident adaptive transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4962–4979, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020. The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6640–6651, Online. Association for Computational Linguistics.

Aviv Slobodkin, Leshem Choshen, and Omri Abend. 2021. Mediators in determining what processing BERT performs first. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 86–93, Online. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

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

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406, Hong Kong, China. Association for Computational Linguistics.

Jue Wang, Ke Chen, Gang Chen, Lidan Shou, and Julian McAuley. 2022. SkipBERT: Efficient inference with shallow layer skipping. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7287–7301, Dublin, Ireland. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, and Lei Li. 2021. A survey on green deep learning. *arXiv:2111.05193v2*.

Sumu Zhao, Damian Pascual, Gino Brunner, and Roger Wattenhofer. 2021. Of Non-Linearity and Commutativity in BERT. In *International Joint Conference on Neural Networks (IJCNN), Virtual-only*.

A. Descriptions of `mat_attn`, `mat_ffn` and `mat_ln1_ln2`

Here we detail the definitions of the mappings $\texttt{mat\_attn}_{\ell\to\ell'}$, $\texttt{mat\_ffn}_{\ell\to\ell'}$ and $\texttt{mat\_ln1\_ln2}_{\ell\to\ell'}$ utilized in §7.

Description of $\texttt{mat\_attn}_{\ell\to\ell'}$. For an input s, let $v_{i_s}^{\ell}$ be the vector at position i_s in the output of $\texttt{attn}_\ell(\texttt{ln1}_\ell(H^{\ell-1}))$. We denote by $A_\ell^{\texttt{attn}} \in \mathbb{R}^{d_h\times d_h}$ the matrix numerically minimizing

$$A \mapsto \sum_{s\in\mathcal{T}} ||A\cdot \texttt{ln1}_\ell(h_{i_s}^{\ell-1}) - v_{i_s}^{\ell}||^2,$$

and define an attention sub-module replacement (Eq. 5) by

$$\texttt{b}_\ell^{\overline{\texttt{attn}}}(h) \coloneqq A_\ell^{\texttt{attn}}\cdot \texttt{ln1}_\ell(h) + h.$$

We then define a mapping between two layers $\ell \to \ell'$ by:

$$\texttt{mat\_attn}_{\ell\to\ell'}(h) \coloneqq$$

$$\texttt{b}_{\ell'}^{\texttt{ffn}}(\texttt{b}_{\ell'}^{\overline{\texttt{attn}}}(\dots(\texttt{b}_{\ell+1}^{\texttt{ffn}}(\texttt{b}_{\ell+1}^{\overline{\texttt{attn}}}(h)))\dots)).$$

Namely, when applying each ℓ''-th block, $\ell < \ell'' \le \ell'$, we replace its attention sub-module $\texttt{attn}_{\ell''}$ by its linear approximation. Importantly, unlike the original attention module, the approximation $\texttt{b}_\ell^{\overline{\texttt{attn}}}$ operates on each position independently, and therefore applying $\texttt{mat\_attn}_{\ell\to\ell'}$ disables any contextualization between the layers ℓ and ℓ'. Note that this is not the case for $\texttt{mat\_ffn}_{\ell\to\ell'}$ and $\texttt{mat\_ln1\_ln2}_{\ell\to\ell'}$, which retain the self-attention sub-modules and operate contextually.

Description of $\texttt{mat\_ffn}_{\ell\to\ell'}$. Let $v_{i_s}^{\ell}$ be the vector at position i_s in the output of $\texttt{ln2}_\ell(\texttt{b}_\ell^{\texttt{attn}}(H^{\ell-1}))$, for a given input s. We denote by $A_\ell^{\texttt{ffn}} \in \mathbb{R}^{d_h\times d_h}$ the matrix numerically minimizing

$$A \mapsto \sum_{s\in\mathcal{T}} ||A\cdot v_{i_s}^{\ell} - \texttt{ffn}_\ell(v_{i_s}^{\ell})||^2,$$

and define a replacement of the feed-forward sub-module $\texttt{b}_\ell^{\texttt{ffn}}$ by

$$\texttt{b}_\ell^{\overline{\texttt{ffn}}}(H) \coloneqq A_\ell^{\texttt{ffn}}\cdot \texttt{ln2}_\ell(H) + H.$$

We then define a mapping between two layers $\ell \to \ell'$ by:

$$\texttt{mat\_ffn}_{\ell\to\ell'}(H) \coloneqq$$

$$\texttt{b}_{\ell'}^{\overline{\texttt{ffn}}}(\texttt{b}_{\ell'}^{\texttt{attn}}(\dots(\texttt{b}_{\ell+1}^{\overline{\texttt{ffn}}}(\texttt{b}_{\ell+1}^{\texttt{attn}}(H))\dots)).$$

Description of $\texttt{mat\_ln1\_ln2}_{\ell\to\ell'}$. Let $v_{i_s}^{\ell}$ be the vector at position i_s in the output of $\texttt{b}_\ell^{\texttt{attn}}(H^{\ell-1})$, for a given input s. We denote by $A_\ell^{\texttt{ln1}} \in \mathbb{R}^{d_h\times d_h}$ the matrix numerically minimizing

$$A \mapsto \sum_{s\in\mathcal{T}} ||A\cdot h_{i_s}^{\ell} - \texttt{ln1}_\ell(h_{i_s}^{\ell})||^2$$

and we denote by $A_\ell^{\texttt{ln2}} \in \mathbb{R}^{d_h\times d_h}$ the matrix numerically minimizing

$$A \mapsto \sum_{s\in\mathcal{T}} ||A\cdot v_{i_s}^{\ell} - \texttt{ln2}_\ell(v_{i_s}^{\ell})||^2.$$

We define a replacement of the block $\texttt{b}_\ell^{\texttt{attn}}$ by

$$\texttt{b}_\ell^{\overline{\texttt{ln1}}}(H) \coloneqq \texttt{attn}_\ell(A_\ell^{\texttt{ln1}}\cdot H) + H \tag{6}$$

and we define a replacement of the block $\texttt{b}_\ell^{\texttt{ffn}}$ by

$$\texttt{b}_\ell^{\overline{\texttt{ln2}}}(H) \coloneqq \texttt{ffn}_\ell(A_\ell^{\texttt{ln2}}\cdot H) + H. \tag{7}$$

We then define a mapping between two layers $\ell \to \ell'$ by:

$$\texttt{mat\_ln1\_ln2}_{\ell\to\ell'}(H) \coloneqq$$

$$\texttt{b}_{\ell'}^{\overline{\texttt{ln2}}}(\texttt{b}_{\ell'}^{\overline{\texttt{ln1}}}(\dots(\texttt{b}_{\ell+1}^{\overline{\texttt{ln2}}}(\texttt{b}_{\ell+1}^{\overline{\texttt{ln1}}}(H))\dots)).$$