Extreme Sub-Updates. To analyze the extreme FFN updates, we first cluster the value vectors to discover high-level trends. We use agglomerative clustering (Müllner, 2011) to learn 10k clusters for each model, based on the cosine distance matrix D, where $D_{(\ell_1,i_1),(\ell_2,i_2)} = 1 - cos(\mathbf{v}_{i_1}^{\ell_1}, \mathbf{v}_{i_2}^{\ell_2})$, $\forall i_1, i_2 \in \{1, \cdots, d_m\}$, $\forall \ell_1, \ell_2 \in \{1, \cdots, L\}$.[6] Then, we search for clusters that are frequently active in extreme updates, by (a) extracting sub-updates where the scores for the top-candidate pass a certain threshold (± 10 for GPT2 and ± 5 for WIKILM), and (b) counting the appearances of each cluster in the layer sub-updates.

In both models, a small set of homogeneous clusters account for the extreme sub-updates shown in Fig. 3, which can be divided into two main groups of value vectors: Vectors in the upper layers that promote *generally unlikely* tokens (e.g. rare tokens), and vectors that are spread over all the layers and promote *common* tokens (e.g. stopwords). These clusters, which cover only a small fraction of the value vectors (1.7% in GPT2 and 1.1% in WIKILM), are mostly active for examples where the input sequence has ≤ 3 tokens or when the target token can be easily inferred from the context (e.g. end-of-sentence period), suggesting that these value vectors might configure "easy" model predictions. More interestingly, the value vectors that promote unlikely tokens can be viewed as *"saturation vectors"*, which propagate the distribution without changing the top tokens. Indeed, these vectors are in the last layers, where often the model already stores its final prediction (Geva et al., 2021).

6 Applications

We leverage our findings for controlled text generation (§6.1) and computation efficiency (§6.2).

6.1 Zero-Shot Toxic Language Suppression

LMs are known to generate toxic, harmful language that damages their usefulness (Bender et al., 2021; McGuffie and Newhouse, 2020; Wallace et al., 2019). We utilize our findings to create a simple, intuitive method for toxic language suppression.

Method. If LMs indeed operate in a promotion mechanism, we reason that we can decrease toxicity by "turning on" non-toxic sub-updates. We find value vectors that promote safe, harmless concepts by extracting the top-tokens in the projections of all the value vectors and either (a) manually searching for vectors that express a coherent set of positive words (e.g. *"safe"* and *"thank"*), or (b) grading the tokens with the Perspective API and selecting non-toxic value vectors (see details in App. A.4). We turn on these value vectors by setting their coefficients to 3, a relatively high value according to Fig. 3. We compare our method with two baselines:

1. Self-Debiasing (SD) (Schick et al., 2021): SD generates a list of undesired words for a given prompt by appending a *self-debiasing input*, which encourages toxic completions, and calculating which tokens are promoted compared to the original prompt. These undesired words' probability are then decreased according to a decay constant λ, which we set to 50 (default).
2. WORDFILTER: We prevent GPT2 from generating words from a list of banned words by setting any logits that would result in a banned word completion to $-\infty$ (Gehman et al., 2020).

Evaluation. We evaluate our method on the challenging subset of REALTOXICPROMPTS (Gehman et al., 2020), a collection of 1,225 prompts that tend to yield extremely toxic completions in LMs, using the Perspective API, which grades text according to six toxicity attributes. A score of > 0.5 indicates a toxic text w.r.t to the attribute. Additionally, we compute perplexity to account for changes in LM performance. We use GPT2 and, following Schick et al. (2021), generate continuations of 20 tokens.

Results. Finding the non-toxic sub-updates manually was intuitive and efficient (taking < 5 minutes). Tab. 5 shows that activation of only 10 value vectors (0.01%) substantially decreases toxicity (↓47%), outperforming both SD (↓37%) and WORDFILTER (↓20%). Moreover, inducing sub-updates that promote "safety" related concepts is more effective than promoting generally non-toxic sub-updates. However, our method resulted in a perplexity increase greater than this induced by SD, though the increase was still relatively small.

6.2 Self-Supervised Early Exit Prediction

The recent success of transformer-based LMs in NLP tasks has resulted in major production cost increases (Schwartz et al., 2020a), and thus has spurred interest in early-exit methods that reduce the incurred costs (Xu et al., 2021). Such methods often use small neural models to determine when to stop the execution process (Schwartz et al., 2020b;

[6] We experimented with $k = 3e^2, 1e^3, 3e^3, 1e^4, 3e^4$, and choose $k = 1e^4$ based on manual inspection.

Model	Toxicity	Severe toxicity	Sexually explicit	Threat	Profanity	Identity attack	PPL
GPT2	58.5%	49.2%	34.1%	16.4%	52.5%	16.8%	21.7
↑ 10 Manual Pick	↓47% 30.8%	↓50% 24.8%	↓40% 20.4%	↓63% 6.0%	↓47% 27.9%	↓48% 8.8%	25.3
↑ 10 API Graded	↓10% 52.7%	↓11% 44%	↓3% 33.2%	↓19% 13.3%	↓9% 47.6%	↓9% 15.3%	23.8
SD	↓37% 37.2%	↓46% 26.4%	↓36% 21.7%	↓52% 7.8%	↓39% 32%	↓50% 8.4%	23.9
WORDFILTER	↓20% 46.9%	↓34% 32.4%	↓36% 21.9%	↓<1% 16.3%	↓38% 32.3%	↓13% 14.7%	-

Table 5: Evaluation results on the challenging subset of REALTOXICPROMPTS, showing the percentage of toxic completions for 6 toxicity attributes, as well as language model perplexity ("PPL").

Elbayad et al., 2020; Hou et al., 2020; Xin et al., 2020, 2021; Li et al., 2021; Schuster et al., 2021).

In this section, we test our hypothesis that dominant FFN sub-updates can signal a *saturation event* (§5.2), to create a simple and effective early exiting method that does not involve any external model training. For the experiments, we use WIKILM, where saturation events occur across all layers (statistics for WIKILM and GPT2 are in App. A.5).

Method. We devise a simple prediction rule based on a nearest-neighbours approach, using 10k validation examples from WIKITEXT-103. First, for every example, we map the top-10 dominant sub-updates at each layer to their corresponding clusters. Then, for every layer ℓ, we split all the sets of clusters at that layer into two sets, T^ℓ and N^ℓ, based on whether saturation occurred or not (e.g., T^5 stores all the sets that were active in a saturation event at layer 5). Given the top-10 clusters of an unseen example at some layer ℓ, we consider a higher overlap with T^ℓ than with $N^{\ell'}$, $\forall \ell' > \ell$ as a signal for early exit. Thus, during inference, we propagate the input example through the layers, and compute at each layer ℓ the intersection size between its top-10 active clusters and each of T^ℓ and $N^{\ell'}$, $\forall \ell' > \ell$. If the average and maximal intersection with T^ℓ exceeds those with $N^{\ell'}$, $\forall \ell' > \ell$, we halt the computation and declare early exiting.[7]

Baselines. We train layer-wise binary classifiers over the representation and FFN updates $\mathbf{x}^\ell$, $\mathbf{o}^\ell$, and $\tilde{\mathbf{x}}^\ell$, using logistic regression. As in our method, the labels are determined according to saturation events in the training data (see App. A.5). During inference, we execute the computation through the layers, and halt according to the layer classifier.

[7]This is a simplification. We split N^ℓ by saturation layers and require a bigger intersection with T^ℓ at all the layers.

Method	Accuracy	Saved Layers
Binary classifiers using $\mathbf{x}^\ell$	94.4±6.4	18.8% 3.0±0.4
Binary classifiers using $\mathbf{o}^\ell$	92.9±5.4	19.4% 3.1±0.3
Binary classifiers using $\tilde{\mathbf{x}}^\ell$	94.4±6.4	18.8% 3.0±0.4
Sub-updates rule	94.1±1.4	20.0% 3.2±0.3

Table 6: Early exit evaluation results on WIKILM.

Evaluation. Each method is evaluated by *accuracy*, i.e., the portion of examples for which exiting at the predicted layer yields the final model prediction, and by *computation efficiency*, measured by the amount of saved layers for examples with correct prediction. We run each method with five random seeds and report the average scores.

Results. Tab. 6 shows that our method obtains a high accuracy of 94.1%, while saving 20% of computation on average without changing the prediction. Moreover, just by observing the dominant FFN sub-updates, it performs on-par with the prediction rules relying on the representation and FFN output vectors. This demonstrates the utility of sub-updates for predicting saturation events, and further supports our hypothesis that FFN updates play a functional role in the prediction (§5.2).

7 Related Work

The lack of interpretability of modern LMs has led to a wide interest in understanding their prediction construction process. Previous works mostly focused on analyzing the evolution of hidden representations across layers (Voita et al., 2019), and probing the model with target tasks (Yang et al., 2020; Clark et al., 2019; Tenney et al., 2019; Saphra and Lopez, 2019). In contrast, our approach aims to interpret the model parameters and their utilization in the prediction process.

More recently, a surge of works have investigated the knowledge captured by the FFN layers (Da et al., 2021; Jiang et al., 2020; Dai et al., 2022;

Yao et al., 2022; Meng et al., 2022; Wallat et al., 2020). These works show that the FFN layers store various types of knowledge, which can be located in specific neurons and edited. Unlike these works, we focus on the FFN outputs and their contribution in the prediction construction process.

Last, our interpretation of FFN outputs as updates to the output distribution relates to recent works that interpreted groups of LM parameters in the discrete vocabulary space (Geva et al., 2021; Khashabi et al., 2022), or viewed the representation as an information stream (Elhage et al., 2021).

8 Conclusions

Understanding the inner workings of transformers is valuable for explainability to end-users, for debugging predictions, for eliminating undesirable behavior, and for understanding the strengths and limitations of NLP models. The FFN is an understudied core component of transformer-based LMs, which we focus on in this work.

We study the FFN output as a linear combination of parameter vectors, termed values, and the mechanism by which these vectors update the token representations. We show that value vectors often encode human-interpretable concepts and that these concepts are promoted in the output distribution.

Our analysis of transformer-based LMs provides a more detailed understanding of their internal prediction process, and suggests new research directions for interpretability, control, and efficiency, at the level of individual vectors.

Limitations

Our study focused on the operation of FFN layers in building model predictions. Future work should further analyze the interplay between these layers and other components in the network, such as attention-heads.

In our analysis, we decomposed the computation of FFN layers into smaller units, corresponding to single value vectors. However, it is possible that value vectors are compositional in the sense that combinations of them may produce new meanings. Still, we argue that analyzing individual value vectors is an important first step, since (a) the space of possible combinations is exponential, and (b) our analysis suggests that aggregation of value vectors is less interpretable than individual value vectors (§4.1). Thus, this approach opens new directions for interpreting the contribution of FFN layers to the prediction process in transformer LMs.

In addition, we chose to examine the broad family of decoder-based, auto-regressive LMs, which have been shown to be extremely effective for many NLP tasks, including few- and zero-shot tasks (Wang et al., 2022). While these models share the same building blocks of all transformer-based LMs, it will be valuable to ensure that our findings still hold for other models, such as encoder-only LMs (e.g. RoBERTa (Liu et al., 2019)) and models trained with different objective functions (e.g. masked language modeling (Devlin et al., 2019)).

Finally, our annotation effort was made for the evaluation of our hypothesis that sub-updates encode human-interpretable concepts. Scaling our annotation protocol would enable a more refined map of the concepts, knowledge and structure captured by LMs. Furthermore, since our concept interpretation approach relies on manual inspection of sets of tokens, its success might depend on the model's tokenization method. In this work, we analyzed models with two different commonly-used tokenizers, and future research could verify our method over other types of tokenizations as well.

Ethics Statement

Our work in understanding the role that single-values play in the inference that transformer-based LMs perform potentially improves their transparency, while also providing useful control applications that save energy (early-exit prediction) and increase model harmlessness (toxic language suppression). It should be made clear that our method for toxic language suppression only reduces the probability of toxic language generation and does not eliminate it. As such, this method (as well as our early-exit method) should not be used in the real world without further work and caution.

More broadly, our work suggests a general approach for modifying LM predictions in particular directions, by changing the weights of FFN sub-updates. While this is useful for mitigating biases, it also has the potential for abuse. It should be made clear that, as in the toxic language suppression application, our approach does not modify the information encoded in LMs, but only changes the intensity in which this information is exposed in the model's predictions. Moreover, our work primarily proposes an interpretation for FFN sub-updates, which also could be used to identify abusive inter-