Normalized Narrow Jump To Conclusions: Normalized Narrow Shortcuts for Parameter Efficient Early Exit Transformer Prediction

Amrit Diggavi Seshadri
Sudarshantech Software
`amrit@sudarshantechsoftware.com`

Abstract

With the size and cost of large transformer-based language models growing, recently, there has been interest in shortcut casting of early transformer hidden-representations to final-representations for cheaper model inference. In particular, shortcutting pre-trained transformers with linear transformations over early layers has been shown to improve precision in early inference. However, for large language models, even this becomes computationally expensive. In this work, we propose *Narrow Jump to Conclusions (NJTC)* and *Normalized Narrow Jump to Conclusions (N-NJTC)* - parameter efficient alternatives to standard linear shortcutting that reduces shortcut parameter count by over 97%. We show that N-NJTC reliably outperforms Identity shortcuts at early stages and offers stable precision from all transformer block levels for GPT-2-XL, Phi3-Mini and Llama2-7B transformer models, demonstrating the viability of more parameter efficient short-cutting approaches.

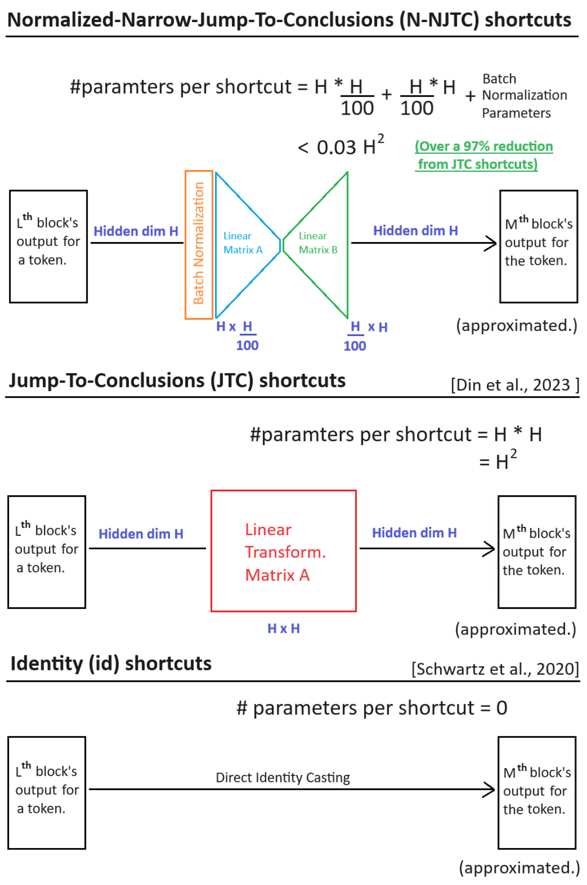

Normalized-Narrow-Jump-To-Conclusions (N-NJTC) shortcuts
#paramters per shortcut = H * H/100 + H/100 * H + Batch Normalization Parameters
< 0.03 H^2
(Over a 97% reduction from JTC shortcuts)
L^th block's output for a token.
Hidden dim H
Batch Normalization
Linear Matrix A
Linear Matrix B
Hidden dim H
M^th block's output for the token.
H x H/100
H/100 x H
(approximated.)
Jump-To-Conclusions (JTC) shortcuts
[Din et al., 2023]
#paramters per shortcut = H * H = H^2
L^th block's output for a token.
Hidden dim H
Linear Transform. Matrix A
Hidden dim H
M^th block's output for the token.
H x H
(approximated.)
Identity (id) shortcuts
[Schwartz et al., 2020]
# parameters per shortcut = 0
L^th block's output for a token.
Direct Identity Casting
M^th block's output for the token.
(approximated.)

Figure 1: Illustration of our N-NJTC short-cut approache in comparison to previous methods.

1 Introduction

Transformer based large language models (Vaswani et al., 2017) stack blocks made up of multi-headed self-attention and feed forward layers sequentially. Modern sophisticated language models stack upwards of 30 such blocks. For example, Phi3-Mini (Abdin et al., 2024) stacks 32 transformer blocks, GPT2-XL (Radford et al., 2019) stacks 48 blocks, and deeper models like Llama-2 70B (Touvron et al., 2023) and GPT-3 (Brown et al., 2020) stack as many as 80 and 96 sequential blocks. However while such stacking typically improves model performance, it also increases the computational costs of inference. More GPU memory is required to store the additional stacked transformer blocks and more time is required to forward-pass inputs sequentially.

There have been attempts to reduce the computational costs of such large language models by short-cutting transformers during inference. In short-cutting, one makes intermediate-predictions from an approximation of the final transformer output - that can be cheaply inferred from intermediate transformer-block outputs at each stage in the forward pass. A decision to 'early-exit' from the forward pass is made once the confidence of these intermediate-predictions reaches a certain pre-set confidence level λ.

Initial methods to short-cutting (Schwartz et al., 2020; Geva et al., 2022) approximated the final transformer output directly by intermediate representations (Identity shortcuts (id) : Figure 1). More recently, (Din et al., 2023) proposed Jump-To-Conclusions (JTC) shortcuts that demonstrate that significant gains in this early-exit inference can be achieved by using a simple linear transformation over token representations to approximate the final transformer output.

However, the JTC linear transformation proposed by (Din et al., 2023) adds an additional H x H parameters for each short-cut inference (for a transformer hidden dimension of H). For deep language models with large hidden dimensions, this too becomes very computationally expensive. For example, if we are to shortcut a Phi3-Mini model, we use a transformer hidden dimension of 3072. So each JTC shortcut roughly requires 9.43 Million new parameters. With 32 transformer-blocks, having a shortcut inference option from each block requires that we train and store 9.43 * 31 > 292 Million new parameters. For larger and deeper models like Llama 2 70B, this number grows to over 5 Billion new shortcut parameters. Clearly, with the size and depth of large transformer models increasing, there is a need to develop more parameter efficient alternatives for short-cutting than the JTC method.

Independent of short-cutting methods, there has also been interest in matrix-decomposition for pre-trained model compression (Lan, 2019; Noach and Goldberg, 2020) and success in low-rank fine-tuning of transformer weights (Hu et al., 2021). LoRA (Hu et al., 2021) in particular has gained traction as a reliable method for parameter efficient fine-tuning - demonstrating that the weight update matrix can be decomposed into low-rank representations to reduce the total number of traninable parameters and reduce costs. However, these prior works focus on increasing efficiency within individual transformer blocks without skipping any block-computations. They do not consider any applications to transformer short-cutting - that skips multiple transformer block computations at a time, and is in itself a method for extreme efficiency.

Taking inspiration from (Noach and Goldberg, 2020) and (Hu et al., 2021), in this work we address the parameter inefficiency of JTC shortcuts (Din et al., 2023).

- We propose Narrow Jump to Conclusions (NJTC) and Normalized Narrow Jump to conclusions (N-NJTC) for shortcutting of transformers - showing that linear shortcuts from early stages can themselves be approximated by low rank representations to achieve over a 97% parameter reduction from JTC shortcuts.
- We show that N-NJTC reliably outperforms Identity shortcuts at early stages and offers stable precision from all transformer block levels for GPT-2-XL, Phi3-Mini and Llama2-7B, demonstrating the viability of more parameter efficient short-cutting approaches.

2 Related Work

As mentioned in Section 1, (Schwartz et al., 2020) was the first to propose shortcuts for early exit transformer prediction. However they make the assumption that all transformer block outputs operate in the same space and use direct identity shortcuts for prediction. (Din et al., 2023) consider the output of different transformer blocks to operate in different representational spaces and recently demonstrated that linear transformation shortcuts significantly improve the performance of early-exit prediction. However, as discussed, they use full $H \times H$ linear matrices and are not very parameter efficient. Independent of these works (Lan, 2019; Noach and Goldberg, 2020) considered matrix-decomposition for pre-trained model compression and (Hu et al., 2021) demonstrated that training low rank matrix decompositions of a weight update matrix approximate good fine-tuning results for transformers with parameter-efficiency. However as mentioned earlier these methods focus on efficiency within fixed transformer-blocks and do not consider any applications to transformer short-cutting - that skips computations of entire blocks at a time, and is itself a method for extreme efficiency.

3 Method

3.1 Narrow Jump To Conclusions (NJTC)

Given a transformer model with hidden dimension H, to approximate a short-cut between its block-outputs at any two levels l and m, given a set of N input sentences S, we forward pass each sentence $s_i \in S$ through the transformer to obtain intermediate representation pairs $\{(h_i^l, h_i^m)\}_{i=1}^N$

after blocks l and m at randomly selected token positions in each s_i. We then fit a simple 2 layer linear neural network made up of matrices $A : H \times \frac{H}{100}$ and $B : \frac{H}{100} \times H$ that takes as input h_i^l and approximates $\hat{h}_i^m$.

$$\hat{h}_i^m = (h_i^l)AB \tag{1}$$

We note in particular that while other model informed-choices for low-rank short-cutting dimensions may be possible, in Eq.1 to standardize our approach to diverse transformers with potentially different hidden dimensions, we consider a fixed low-rank reduction to 1% of the transformer hidden dimension size ($\frac{H}{100}$).

We fit the two matrices A and B jointly using gradient descent to minimize the mean squared error loss L_{lm} between the approximated representations $\hat{h}_i^m$ and the transformer block outputs h_i^m.

$$L_{lm} = \frac{1}{N}\sum_{i=1}^{N} ||\hat{h}_i^m - h_i^m||^2 \tag{2}$$

The hidden representation of each token in a sentence at level l is passed through A and B to obtain approximations of the hidden representations at level m. As a result of this low rank matrix decomposition, each NJTC shortcut uses $2 * (H \times \frac{H}{100}) = 0.02H^2$ parameters: Only 2% the number of parameters of a JTC shortcut.

3.2 Normalized Narrow Jump To Conclusions (N-NJTC)

We note that our NJTC method can be viewed as a special form of a linear denoising auto-encoder - where, the ‘corrupted input’ is a transformer’s early block hidden representation h_i^l and the restoration target is a block’s output h_i^m further down the forward pass. Linear autoencoders usually learn latent dimensions that maximize feature variance and are preceeded by normalization along the batch-dimension to avoid any bias towards naturally high-variance features. Motivated by this comparison, we propose a normalized version of NJTC where we add a batch normalization layer before AB (Fig. 1). Batch Normalization adds an additional 4H parameters for each shortcut. For hidden dimension $H > 400$, this is less than $0.01H^2$. As all transformer models use a hidden dimension larger than 400, we find that N-NJTC uses less than 3% the number of parameters of a JTC shortcut - offering over a 97% parameter reduction.

4 Experiments

We test our shortcuts on GPT2-XL (Radford et al., 2019) which consists of 48 transformer blocks, hidden dimension of 1600, and a total of 1.5 Billion model parameters; on the larger Phi3-Mini (Abdin et al., 2024) which uses 32 transformer blocks, has hidden dimension 3072, and has a total of 3.8 Billion parameters; and on the even larger Llama2-7B (Touvron et al., 2023) that uses 32 transformer blocks, has hidden dimension 4096 and uses a total of 7 Billion parameters. The low-rank dimensions $\lfloor \frac{H}{100} \rfloor$ that we use for our NJTC and N-NJTC shortcuts for GPT2-XL, Phi3-Mini and Llama2-7B are 16 and 30 and 40 respectively.

Data: Following the approach taken by (Din et al., 2023), we sample random sentences from Wikipedia, collecting 9,000 train sentences and 3000 validation sentences - each of which are highly diverse, written by different authors on varied topics. As explained in Section 3.1, each sentence is forward passed through a given transformer model and random token position representations are selected across all hidden representations to train and evaluate shortcuts.

4.1 Quality of Shortcut Approximations

We first examine the degree of correlation between true transformer block outputs and their shortcut approximations for each shortcut type. For this purpose, we compute the coordinate averaged r2 scores between true transformer Block M outputs and corresponding shortcut jump approximations made from Block N outputs to Block M. Figure 2 shows heatmaps of these scores across all transformer block levels for id, JTC, NJTC and N-NJTC shortcuts for GP2-XL, Phi3-Mini and Llama2-7B models respectively.

For id and JTC shortcuts, as one would expect, correlation of approximations seems to worsen as jump distance increases. That is, we always achieve better correlated approximations by making a shortcut jump from Block N to Block $(N + 1)$ than we could achieve by making a jump to a later Block $(N + 2)$. Interestingly, for NJTC and N-NJTC shortcuts, that is not the case. As

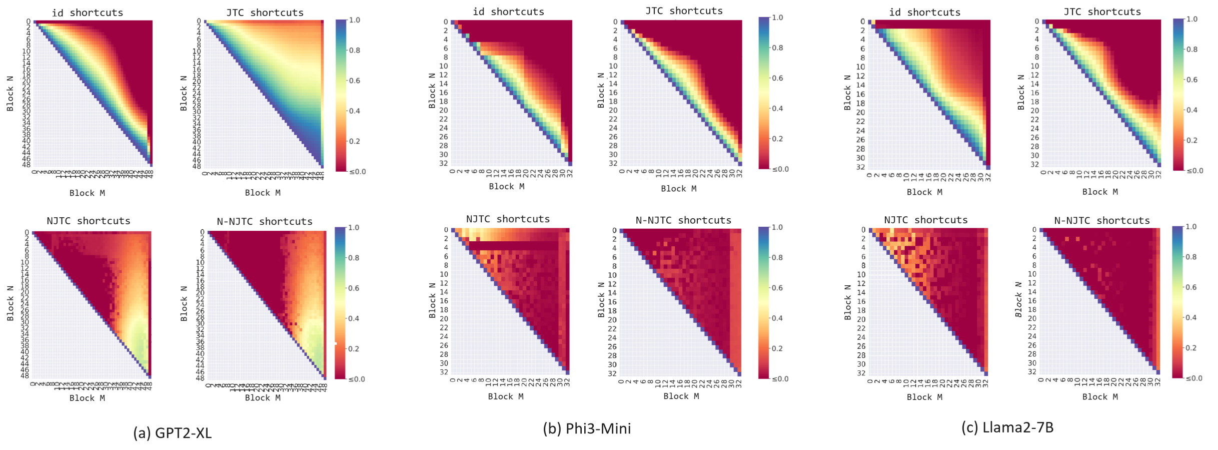


Figure 2: Coordinate-averaged r2-scores (↑) between real outputs from transformer Block M and shortcut jump approximations to M from N for different shortcuts types for (a) GPT2-XL, (b) Phi3-Mini and (c) Llama2-7B.

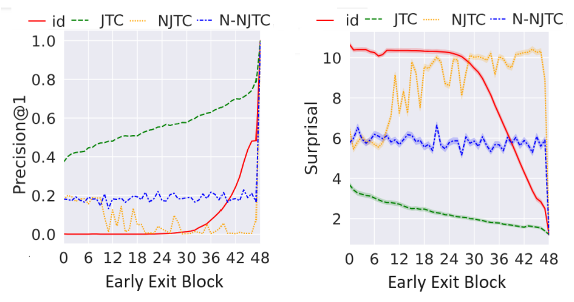
shown in Figure 2, with some exceptions, we typically achieve better correlated approximations by jumping from any intermediate block N directly to the final few blocks output than we could achieve by making a smaller jump from block N to $(N + 1)$ at earlier stages. This is an important finding as making jumps to the final block output is all that we really care about for early exit transformer prediction. We are happy to sacrifice intermediate jump quality to improve parameter efficiency of our shortcuts, provided that jumps to the final block outputs are still well correlated with the true final outputs. In this context, we note that N-NJTC shortcuts usually provide better correlated approximation in the final blocks than NJTC shortcuts can.

4.2 Quality of Shortcut Predictions

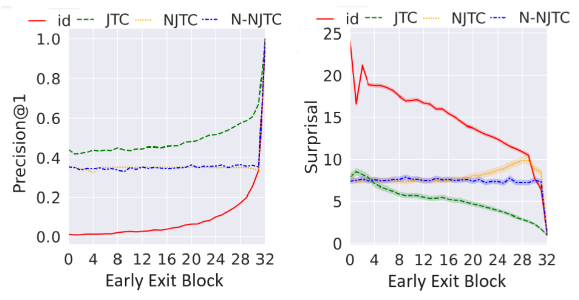
We next consider the quality of shortcut approximations for next token predictions obtained by shortcut jumping to the final transformer block output from each intermediate block. Following the approach taken by (Din et al., 2023), we compute Precision by assigning a score of 1 if the most-likely token from the shortcut predicted distribution matches the most-likely token from the true final block output distribution and 0 otherwise; and compute Surprisal by measuring the negative log-likelihood of the true block most likely output token according to the shortcut predicted distribution. Figure 3 shows these Precision and Surprisal scores achieved by id, JTC, NJTC and N-NJTC shortcuts when making an early exit from each transformer block for GPT-2XL, Phi3-Mini and Llama2-7B models respectively.

As expected, reducing parameter count by 97%, NJTC and N-NJTC shortcuts record lower precision and higher surprisal scores than JTC shortcuts for all models. However, their behaviour is still unexpected and interesting. Our main contribution is the finding that despite the drastic 97% reduction in parameter count from JTC shortcuts, N-NJTC is still able to reliably outperform Identity shortcuts (id) at early transformer model stages. As shown in Figure 3, N-NJTC acheives steady and non-fluctuating scores, recording higher precision and lower surprisal than Identity shortcuts (id) upto at least 50% of a model's total block depth for all three GPT-2XL, Phi3-Mini and Llama2-7B models. This is not a guaranteed finding. With such a large reduction in paramter count, an intutive expectation is that precision and surprisal would record very poor values or fluctuate greatly and collapse quickly. This does infact happen for NJTC in GPT2-XL prediction (Figure 3). However N-NJTC solves this problem and remains steady.

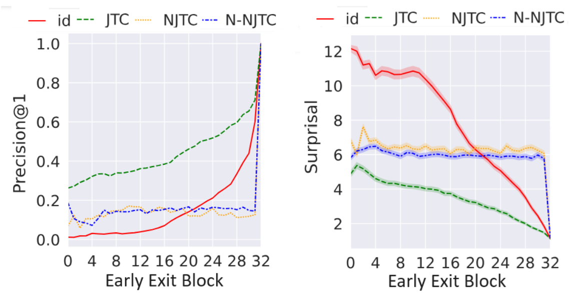
To further highlight the surprising nature of trends being observed in a concrete example, consider the Phi3Mini model and a jump from layer 4 to 32. That shortcut jump skips 87.5% of transformer block computations, i.e > 3 Billion parameter computations. We would expect an id shortcut that uses hidden dimension of size 3072 to perform badly for the missed computation. Intuitively, we would not expect any improvements to result from compressing that early latent vector down from hidden dimension 3072 to 30 and then decompressing it. We would expect a collapse in performance - resulting from a loss of the



(a) GPT2-XL



(b) Phi3-Mini



(c) Llama2-7B

Figure 3: Precision (↑) and Surprisal (↓) achieved by different shortcut types for early exit prediction for (a) GPT-2-XL, (b) Phi3-Mini and (c) Llama2-7B.

information that the early transformer output had. However, contrary to that intuition, we find that for our N-NJTC method, not only is there no collapse in performance, but a significant improvement in early transformer block stages - (Figure 3.b). Further, this surprising trend of improvement holds steady up to at least 50% of model-depth across diverse transformer models – GPT2-XL, Phi3Mini, Llama2-7B that vary in size, vary in structure, vary in creators, and vary in training data. The main contribution of our paper is the surprising finding that the drastic parameter efficiency of our N-NJTC method is indeed viable for transformer short-cutting – given that short-cutting is already a method for extreme computational efficiency itself.

5 Practicality and Future Work

In terms of immediate practically, we highlight that our N-NJTC method can offer immediate cost savings in settings where one exits the transformer stack early (before executing 50% of the model's total block depth) and when computation overhead from a full JTC shortcut is substantial, while the performance from Identity shortcuts is unacceptable. In terms of future work on the other hand, we highlight that to our knowledge, we are the first to examine the problem of parameter inefficiency in JTC shortcuts, and the first to consider parameter efficient shortcutting alternatives – we register our surprise that such drastic improvements up to 97% less costly are indeed viable in early stages, and expect that this observation will spur future interest in building more variants of parameter efficient transformer shortcutting approaches.

6 Conclusion

In this work, we proposed the *Narrow Jump to Conclusions (NJTC)* and *Normalized Narrow Jump to Conclusions (N-NJTC)* methods for parameter efficient shortcutting of transformer models. We showed that linear shortcuts from early stages can themselves be approximated by low rank representations to achieve over a 97% parameter reduction from JTC shortcuts. We applied our NJTC and N-NJTC methods to GPT-2-XL, Phi3-Mini and Llama2-7B transformer models and showed that N-NJTC reliably outperforms Identity shortcuts at early transformer model stages while also offering stable precision and surprisal from all transformer block levels, demonstrating the viability of more parameter efficient short-cutting methods than JTC.

7 Limitations

Notably, as mentioned in Section 4.2, our NJTC method collapses for GPT2-XL and while N-NJTC solves this problem, NJTC and N-NJTC both achieve worse precision and surprisal scores than JTC shortcuts for all models, and are outperformed by Identity shorotcuts in late-block shortcutting (Figure 3). We note however that these limitations are acceptable in exchange for the 97% reduction in parameter count our N-NJTC method offers while outperforming Identity shortcuts at early transformer model stages. We note that shortcutting of transformers in general can cause unexpected model behaviour and caution that any shortcut approximations be tested for safety.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2023. Jump to conclusions: Short-cutting transformers with linear transformations. *arXiv preprint arXiv:2303.09435*.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Z Lan. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Matan Ben Noach and Yoav Goldberg. 2020. Compressing pre-trained language models by matrix decomposition. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 884–889.

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

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A Smith. 2020. The right tool for the job: Matching model and instance complexities. *arXiv preprint arXiv:2004.07453*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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.