# Tandem Transformers for Inference Efficient LLMs

**Aishwarya P S** [1] **Pranav Ajit Nair** [1] **Yashas Samaga** [1] **Toby Boyd** [1] **Sanjiv Kumar** [2]
**Prateek Jain** [*1] **Praneeth Netrapalli** [*1]

## Abstract

The autoregressive nature of conventional large language models (LLMs) inherently limits inference speed, as tokens are generated sequentially. While speculative (Leviathan et al., 2023) and parallel (Stern et al., 2018) decoding techniques attempt to mitigate this, they face limitations: either relying on less accurate smaller models for generation or failing to fully leverage the base LLM's representations.

We introduce a novel architecture, Tandem Transformers, to address these issues. This architecture uniquely combines (1) a small autoregressive model and (2) a large model operating in block mode (processing multiple tokens simultaneously). The small model's predictive accuracy is substantially enhanced by granting it attention to the large model's richer representations. On the PaLM2 pretraining dataset, a Tandem of PaLM2-Bison and PaLM2-Gecko demonstrates a 3.3% improvement in next-token prediction accuracy over a standalone PaLM2-Gecko, offering a 1.16x speedup compared to a PaLM2-Otter model with comparable downstream performance. We further incorporate the Tandem model within the speculative decoding (SPEED) framework where the large model validates tokens from the small model. This ensures that the Tandem of PaLM2-Bison and PaLM2-Gecko achieves substantial speedup (around $1.14\times$ faster than using vanilla PaLM2-Gecko in SPEED) while maintaining identical downstream task accuracy.

[*]Equal contribution [1]Google DeepMind [2]Google Research, New York City. Correspondence to: Aishwarya P S <aishwaryaps@google.com>, Praneeth Netrapalli <pnetrapalli@google.com>.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

## 1. Introduction

Despite significant advancements in inference optimization techniques (Leviathan et al., 2023; Du et al., 2022; Liu et al., 2023), the widespread deployment of very large language models (LLMs) remains hindered by their substantial computational costs. A key factor contributing to high inference latency is the autoregressive generation process, where tokens are produced sequentially. This inherent limitation restricts the full utilization of ML accelerators (GPUs/TPUs), which are optimized for matrix-matrix multiplications rather than the matrix-vector operations prevalent in LLMs. Consequently, prompt processing (where all tokens are handled simultaneously) is significantly more efficient than autoregressive response generation.

On the other hand, it is not well understood how much capacity is required to understand the prompt/query/prefill (natural language understanding aka NLU) vs the capacity required to generate a response (natural language generation aka NLG). Current decoder-only LLM architectures tightly couple both these tasks.

**Tandem Transformers.** In this work, we investigate this fundamental question from an efficiency perspective. We propose Tandem Transformers, a novel architecture that allocates significantly more model capacity to prefill processing (NLU) compared to response generation (NLG). Our goal is to understand whether high-quality response generation can be maintained under this design. Concretely, Tandem Transformers consists of two models – a small model $\mathcal{M}_S$ and a large model $\mathcal{M}_L$, where:

1. $\mathcal{M}_L$ processes the prompt/query.
2. $\mathcal{M}_S$ generates the first $\gamma$ tokens (called a *block*) autoregressively, while attending to the prompt/query representations generated by $\mathcal{M}_L$.
3. $\mathcal{M}_L$ processes the $\gamma$ tokens generated by $\mathcal{M}_S$ together (i.e., in a non-autoregressive fashion) and computes their representations.
4. $\mathcal{M}_S$ then generates the next $\gamma$ tokens autoregressively, while attending to representations of all tokens until the previous prefill *block* generated by $\mathcal{M}_L$.
5. This process is repeated until the response generation is complete.

**Tandem Transformer Training.** A projection layer is introduced to align the higher-dimensional representation space of $\mathcal{M}_L$ with that of $\mathcal{M}_S$. For efficiency, we initialize $\mathcal{M}_L$ and $\mathcal{M}_S$ as independently trained, decoder-only models.

Experiments with Tandem (PaLM2-Bison, PaLM2-Gecko) (where PaLM2-Gecko $<$ PaLM2-Otter $<$ PaLM2-Bison, in terms of model size) demonstrate that the capacity needed for NLU vs NLG aspects of LLMs can indeed be decoupled, leading to a more efficient architecture without significant accuracy loss. Evaluation on benchmark datasets show that Tandem (PaLM2-Bison, PaLM2-Gecko) with block length $\gamma = 3$ is substantially more accurate than PaLM2-Gecko, and comparable to PaLM2-Otter, while achieving approximately $1.16\times$ lower inference latency than PaLM2-Otter. For example, on SuperGLUE (Wang et al., 2019), the Tandem model is $3\%$ less accurate than PaLM2-Bison, $16\%$ more accurate than PaLM2-Gecko and $0.2\%$ less accurate than PaLM2-Otter, with $1.16\times$ speedup over PaLM2-Otter.

**Encoder-Decoder.** In contrast to an encoder-decoder architecture which would only process query/prefix through an encoder and then generate the entire response through a decoder, Tandem is able to generate only block-size $\gamma$ (say $= 3$) tokens through the secondary model $\mathcal{M}_S$ and then refresh the entire prefill representations using primary model $\mathcal{M}_L$ which is critical to maintaining high accuracy. That is, by setting $\gamma = 0$, Tandem can mimic decoder-only $\mathcal{M}_L$ model while setting $\gamma \to \infty$ leads to decoder-only $\mathcal{M}_S$ model.

**Tandem + SPEED.** For applications requiring output identical to the primary model, we propose Tandem + SPEED. The speculative decoding (SPEED) framework (Leviathan et al., 2023) leverages the small model $\mathcal{M}_S$ in Tandem to generate draft tokens, which are then verified by the large model $\mathcal{M}_L$. Crucially, the ability of $\mathcal{M}_S$ in Tandem to attend to $\mathcal{M}_L$'s representations significantly improves draft quality, reducing verification overhead compared to standard SPEED. For example, on the Reddit Posts dataset, using the $\mathcal{M}_S$ in Tandem as the drafter model in SPEED leads to about $11.24\%$ higher per-block acceptance rate compared to a vanilla secondary model. Finally, we show that Tandem Transformers can be further improved using logit distillation and their efficacy within SPEED can be improved using an adaptive block length parameter.

**Contrast with Parallel Decoding and Distillation.** Recently multiple speculative or parallel decoding style techniques have been proposed in the literature (Leviathan et al., 2023; Kim et al., 2023; Stern et al., 2018). These techniques attempt to generate a draft of tokens using a relatively inexpensive drafter model. Parallel decoding attempts to generate multiple drafter tokens in parallel by learning classifiers on top of output of primary model $\mathcal{M}_L$ while speculative decoding could provide significantly better drafts by using a small, but auto regressive model. In contrast, Tandem is a *stand alone* model on its own and doesn't natively require verification by $\mathcal{M}_L$ to generate reasonable outputs (see benchmark numbers in Table 3). Furthermore, Tandem + SPEED is able to use representations of $\mathcal{M}_L$ while still generating tokens autoregressively, which is able to provide overall much better tradeoff in terms of token quality vs model latency for the drafter. Finally, recent works have also shown the efficacy of logit distillation for training better drafter models within SPEED (Zhou et al., 2023). Our approach is complementary, and can be combined with distillation.

**Empirical Results for Tandem + SPEED.** Finally, we conduct extensive latency evaluation on TPUv5e for both standa alone and SPEED versions of Tandem (PaLM2-Bison, PaLM2-Gecko) with PaLM2-Bison and PaLM2-Gecko being the primary $\mathcal{M}_L$ and secondary $\mathcal{M}_S$ model, respectively. In particular, on multiple datasets, we observe that Tandem + SPEED with distillation can be at least $2.19\times$ faster than the baseline PaLM2-Bison model while ensuring same output quality. Furthermore, compared to standard SPEED with $\mathcal{M}_S$ being secondary model, our model is $1.11\times$ to $1.17\times$ faster. An adaptive block length in SPEED further helps reduce Tandem's latency by $1.04\times$ to $1.09\times$ on multiple datasets. Finally, we demonstrate that our results also hold for practical settings like batch-size $> 1$.

**Contributions.** In summary, following are the key contributions of the work:

1. Tandem architecture: A novel architecture to disaggregate prompt/prefill processing capacity from response generation.
2. Tandem + SPEED: Improved speculative decoding leveraging Tandem's superior drafting for guaranteed output equivalence with lower latency.
3. Adaptive Block Length: Enhances Tandem + SPEED by dynamically adjusting drafted token count.
4. TPUv5e evaluation: End-to-end evaluation on TPUv5e with PaLM2-Bison being the primary model. A distilled Tandem + SPEED is $2.4\times$ faster compared to vanilla PaLM2-Bison model and $1.11 - 1.17\times$ faster compared to distilled $\mathcal{M}_S$ + SPEED (Leviathan et al., 2023) applied in the same setting.

**Outline of the Paper.** The rest of the paper is organized as follows. We briefly review related work in Section 2. In Section 3, we present the main ideas and the design of Tandem Transformers architecture. Section 4 presents the experimental results on Tandem Transformers. We then conclude with some future directions in Section 5.

## 2. Related Work

**Encoder-Decoder models.** Encoder-decoder Transformer architectures are widely used for specific tasks such as machine translation (Vaswani et al., 2017). Given the computational inefficiency of autoregressive decoding, several works have explored using a large encoder with a small decoder. Our work can be seen as extending these ideas to use an encoder-decoder model for the decoder itself.

**Mixture of Experts (MoE)/Sparsity based Approaches.** Mixture of experts (Du et al., 2022) and sparsity based approaches (Li et al., 2022) have also been studied for optimizing inference cost of LLMs. However these approaches are complementary to the approaches proposed in our paper. For example, either or both the large model $\mathcal{M}_L$ and small model $\mathcal{M}_S$ can be an MoE or sparse model.

**Distillation.** Since the seminal paper (Hinton et al., 2015), distilling the knowledge of a large model to a smaller model by using the logits of large model as a training target has been widely used in several settings. Our work can be seen as a more general version of distillation for Transformers, where the small model can directly refer to large model representations for tokens from previous blocks. Furthermore, our experiments (see Section 4) show that our techniques are complementary to logit distillation, and provide additional gains on top of vanilla logit distillation.

**Speculative Decoding (SPEED).** Speculative decoding (Leviathan et al., 2023; Kim et al., 2023) is a framework to reduce inference latency of LLMs without affecting their quality, which has shown substantial improvements in LLM inference. We demonstrate that Tandem Transformers can be used within the SPEED framework, improving the efficacy of SPEED. Multiple drafters have been explored in the context of SPEED, Speculative Decoding (Leviathan et al., 2023) uses a standalone drafter, REST (He et al., 2023) use a retrieval based drafter, DistilSpec (Zhou et al., 2023) uses a distillation based drafter, MEDUSA (Cai et al., 2024) uses $k$ MLP heads on top of the primary model's output representations to predict the next $k$ tokens in parallel for drafting. As of now distillation based drafters seem to perform the best. As we demonstrate in Section 4, Tandem is able to provide a significantly more powerful drafter, thus providing better draft of tokens leading to lower latency.

## 3. Tandem Transformers

In this section, we will describe Tandem Transformers architecture, it's training and inference.

**Standard (Decoder) Transformer.** Given a sequence $t_1, t_2, \cdots, t_S$ of $S$ tokens as inputs, where $t_i$ corresponds to the $i^{\text{th}}$ token id, a standard decoder Transformer with $L$ layers executes as follows:

$$\begin{aligned} \widetilde{x}_i^{(j+1)} &= \text{Atn}^{(j+1)}(x_i^{(j)} | x_{\leq i}^{(j)}) \\ x_i^{(j+1)} &= \text{FF}^{(j+1)}(\widetilde{x}_i^{(j+1)}) \qquad \text{for } j = 0, \cdots, L-1, \end{aligned} \tag{1}$$

where $x_i^{(0)} = \text{Emb}(t_i)$ is the embedding of $t_i$, $x_i^{(j)}$ is the representation after the $j^{\text{th}}$ layer and $\text{Atn}^{(j)}(\cdot|\cdot)$ and $\text{FF}^{(j)}(\cdot)$ are the $j^{\text{th}}$ attention and feedforward layers respectively (Vaswani et al., 2017). Note that the attention is purely causal (i.e., the $i^{\text{th}}$ token attends only tokens $t_k$ for $k \leq i$) since we are considering a decoder-only Transformer.

**Tandem Transformer.** A Tandem Transformer model comprises of a primary model $\mathcal{M}_L$ and a secondary model $\mathcal{M}_S$. Typically, $\text{SIZEOF}(\mathcal{M}_L) \gg \text{SIZEOF}(\mathcal{M}_S)$. Given a sequence of tokens $t_1, t_2, \cdots, t_S$ as inputs, the primary model $\mathcal{M}_L$ processes these tokens just like a standard (decoder) Transformer (1).

Let $\gamma$ be the block length parameter, and $L_S$ and $L_L$ be the number of layers of the secondary model and primary model, respectively. Let $\ell : [L_S] \to [L_L]$ be a layer assignment function from secondary model to primary model. The secondary model attends to the primary model's representations for all tokens from the previous blocks. More formally, we have:

$$\begin{aligned} \widehat{y}_i^{(j)} &= \text{FF}_{\text{Tandem}}^{(j)}(x_i^{(\ell(j))}) \\ \widetilde{y}_i^{(j+1)} &= \text{Atn}_{\text{S}}^{(j+1)}(y_i^{(j)} | \widehat{y}_{\leq k}^{(j)}, y_{[k+1,i]}^{(j)}) \text{ where } k = \lfloor \frac{i}{\gamma} \rfloor * \gamma \\ y_i^{(j+1)} &= \text{FF}_{\text{S}}^{(j+1)}(\widetilde{y}_i^{(j+1)}) \qquad \text{for } j = 0, \cdots, L_S - 1, \end{aligned} \tag{2}$$

where $x_i^{(j)}$ and $y_i^{(j)}$ denote the $j^{\text{th}}$ layer representation of the $i^{\text{th}}$ token under $\mathcal{M}_L$ and $\mathcal{M}_S$ respectively, $\text{FF}_{\text{Tandem}}^{(j)}(\cdot)$ denotes a feedforward layer that converts the representation $x_i^{(\ell(j))}$ of the $i^{\text{th}}$ token from the $\ell(j)^{\text{th}}$ layer of the primary model, to a representation $\widehat{y}_i^{(j)}$ of the same $i^{\text{th}}$ token for the $j^{\text{th}}$ layer of the secondary model, and $\text{Atn}_{\text{S}}^{(j)}(\cdot|\cdot)$ and $\text{FF}_{\text{S}}^{(j)}(\cdot)$ denote the attention and feedforward blocks respectively in the $j^{\text{th}}$ layer of the secondary model $\mathcal{M}_S$. The final output of the Tandem model is $y^{(L_S)}$. We note that the primary and the secondary model can vary in almost all scale parameters such as representation dimensions, expansion factors of feedforward layers, number of attention heads, etc. as well as whether the attention is multi-head or multi-query, etc. In all of our experiments, we take $\text{FF}_{\text{Tandem}}^{(\cdot)}(j)$ to be linear projection layers.

**Training.** Given a block length parameter $\gamma$, we partition the training sequence into blocks, each consisting of $\gamma$ consecutive tokens. Consider the autoregressive prediction of the $j^{\text{th}}$ token (for some $j \leq \gamma$) within the $i^{\text{th}}$ block. The

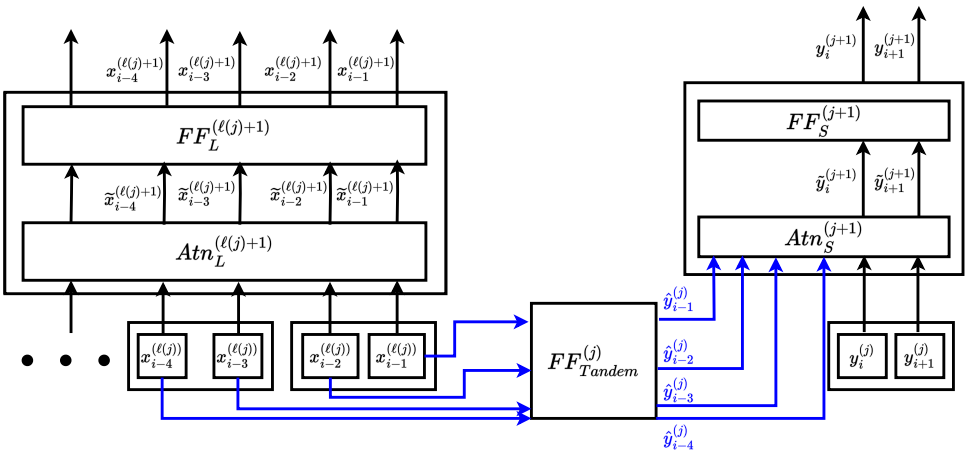


*Figure 1.* Training of Tandem Transformers with a block length $\gamma = 2$. $\text{Atn}_L^{(\ell(j)+1)}$ and $\text{FF}_L^{(\ell(j)+1)}$ denote the attention and feedforward blocks in the $(\ell(j)+1)^{\text{th}}$ layer of $\mathcal{M}_L$, while $\text{Atn}_L^{(j+1)}$ and $\text{FF}_L^{(j+1)}$ denote those of $(j+1)^{\text{th}}$ layer of $\mathcal{M}_S$. $\mathcal{M}_L$ processes the tokens as a standard decoder Transformer. $\mathcal{M}_S$ on the other hand processes the tokens in the $\left(\frac{i}{\gamma}\right)^{\text{th}}$ block using its own representations $y_i^{(j)}$ and $y_{i+1}^{(j)}$, but while attending to the representations of all tokens from the previous block from the $(\ell(j)+1)^{\text{th}}$ layer of $\mathcal{M}_L$ passed through a feedforward layer $\text{FF}_{\text{Tandem}}^{(j)}$.

input to the secondary model $\mathcal{M}_S$ is the previous token. Crucially, within the attention blocks of $\mathcal{M}_S$:

- Key/value pairs for all tokens up to the $j^{\text{th}}$ token in the *current block* are computed by $\mathcal{M}_S$ itself.
- Key/value pairs for tokens in previous blocks are computed by the primary model $\mathcal{M}_L$. A projection/Tandem feedforward layer then aligns the representational dimensions from $\mathcal{M}_L$ to $\mathcal{M}_S$, as described in Equation (2).

We explore multiple training configurations for Tandem Transformers:

- **Primary Model Frozen.** Only the secondary model parameters $\mathcal{M}_S$ and the Tandem feedforward layer $\text{FF}_S^{(j)}$ are updated. Loss is applied solely to the secondary model's output $y^{(L_S)}$ (Equation (2)).
- **Both Models Trained, Loss on Secondary Outputs.** Similar to the above, loss is applied to the secondary model's output. However, both $\mathcal{M}_L$ and $\mathcal{M}_S$, along with $\text{FF}_S^{(j)}$ are trained.
- **Both Models Trained, Loss on Both Outputs.** The combined loss incorporates both the primary model's outputs $x^{(L_L)}$ and the secondary model's outputs $y^{(L_S)}$.

For training efficiency, we initialize the primary and secondary models with high quality pretrained checkpoints, and then continue pretraining the Tandem architecture for a small number of additional steps. In particular, we use the pretrained PaLM2-Bison and PaLM2-Gecko checkpoints to initialize $\mathcal{M}_L$ and $\mathcal{M}_S$ respectively. In this setting, we found that **Primary Model Frozen** approach provides the best accuracy. Our Tandem-CE model is obtained by using cross entropy (CE) loss on the output of the secondary model as described above.

Tandem-Distil: To further enhance $\mathcal{M}_S$'s quality, we apply a distillation loss on its predictions, using the logits of the pretrained $\mathcal{M}_L$ as targets with CE loss. This aligns naturally with the Tandem architecture, as $\mathcal{M}_S$ already incorporates representations from $\mathcal{M}_L$.

The Tandem-Distil model follows a two stage training setup, where initially it is trained to minimize the CE loss with respect to the ground truth labels, and in the second stage a weighing factor of $\lambda = 0.5$ is used to balance the CE loss with respect to ground truth labels and the CE logit distillation loss with respect to the outputs of the PaLM2-Bison model. We note that Tandem-Distil in general performs better than Tandem-CE.

**Inference.** The inference process begins with the primary model ($\mathcal{M}_L$) processing the prompt and generating representations for all prompt tokens. The secondary model ($\mathcal{M}_S$) then autoregressively generates the first block of $\gamma$ response tokens. Crucially, $\mathcal{M}_S$ attends to the primary model's representations, aligned via the projection layer.

Once the first response block is generated, the primary model ($\mathcal{M}_L$) processes these tokens and computes their representations. We consider two inference configurations:

- **Representation Generation + Token Prediction (Figure 2).** $\mathcal{M}_L$ additionally predicts the next token.
- **Representation Generation Only (Appendix B, Figure 3).** $\mathcal{M}_L$ solely generates representations for the response block.

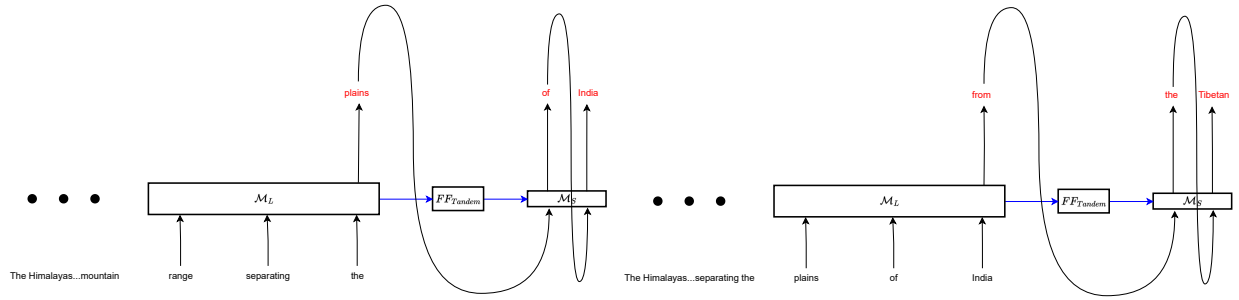


*Figure 2.* Inference of Tandem Transformers *with* free token from the primary model $\mathcal{M}_L$. (left) First block prediction. (right) Second block prediction. Given the query *The Himalayas are a mountain range separating the*, $\mathcal{M}_L$ first processes this query and produces the first response token plains. When we use this prediction from $\mathcal{M}_L$, this is directly fed as an input to the secondary model $\mathcal{M}_S$, which autoregressively produces of India for the first block with $\gamma = 2$. In the second block, the entire response from the first block plains of India is fed to the primary model $\mathcal{M}_L$, which again produces the next response token from, and then the secondary model $\mathcal{M}_S$ produces the next two tokens of the block the Tibetan autoregressively. The eventual output of the model will be plains of India from the Tibetan ....

In both configurations, the representations generated by $\mathcal{M}_L$ are used by the secondary model ($\mathcal{M}_S$) to generate the subsequent block of $\gamma$ response tokens. Also note that, as in training, $\mathcal{M}_S$ attends to its own representations for all previous tokens within the current block.

To disaggregate query and response generation, we use **Representation Generation Only** for processing the input query/prefix. However, for subsequent blocks where the prefill (query+generated response till this point) is processed, we use **Representation Generation + Token Prediction** from $\mathcal{M}_L$.

Depending on the training protocol – specifically, whether primary model outputs are reliable – we may optionally allow the primary model ($\mathcal{M}_L$) to generate the first token of the subsequent block (processing $\gamma + 1$ tokens). Crucially, in this scenario, we must ensure the following: the keys and values associated with the next block's first token, computed by $\mathcal{M}_L$, are not overwritten when the secondary model ($\mathcal{M}_S$) executes its attention layers.

**Inference-Time Block Length Flexibility.** While we train Tandem Transformers with a fixed block length $\gamma$, the architecture supports arbitrary $\gamma$ values during inference. Larger $\gamma$ values generally improve efficiency by maximizing the primary model's ($\mathcal{M}_L$) utilization of accelerator hardware. Although Tandem is trained with a fixed $\gamma$, in SPEED evaluations we find that the optimal $\gamma$ is often much larger, indicating the robustness of Tandem to changes in $\gamma$ at inference time.

### 3.1. Tandem + SPEED: Tandem in the Speculative Decoding Framework

SPEED mitigates the inefficiency of autoregressive generation using a smaller drafter/secondary model to generate tokens and a larger verifier/primary model to confirm them. SPEED guarantees output quality matching the verifier, but its efficacy hinges on the drafter's ability to generate long, accurate draft sequences. Tandem Transformers are uniquely suited for this framework, with our secondary model $\mathcal{M}_S$ acting as the "drafter" and primary model $\mathcal{M}_L$ acting as the "verifier".

Given a Tandem model, we use $\mathcal{M}_L$ to process the query/prefix and generate representations for them. $\mathcal{M}_S$ uses these and produces a draft for the first $\gamma$ tokens autoregressively. $\mathcal{M}_L$ then verifies this entire block simultaneously and identifies the first location $i$ where the draft token is deemed incorrect by $\mathcal{M}_L$ ($i = \gamma + 1$, if all the draft tokens are verified successfully). We take the output of the large model for the $i^{\text{th}}$ token, and the small model $\mathcal{M}_S$ then continues to generate draft tokens from the $(i+1)^{\text{th}}$ position onwards, while using the representations of *all the previous tokens* from the large model $\mathcal{M}_L$. This process continues until a full response is generated.

The above process can be generalized to the setting, where we generate multiple full responses for the same query, we refer to it as num-samples, for example to eventually rank these responses and select the "best" response (Mudgal et al., 2023). In this case, the location of the rejected token can vary across the different samples being generated.

Similarly, the above approach generalizes to larger batch sizes as well, when we are simultaneously processing multiple queries together. Practical systems potentially use both num-samples and batch-size to be $> 1$ but latency gains for Tandem + SPEED depend on overall batch-size which is num-samples $\times$ $batch_size$. So, for simplicity we focus only on num-samples $> 1$ and fix batch-size to be 1[1].

[1]Note that it is more challenging to obtain latency improvements with increasing num-samples, compared to that in batch size since, even without any of these optimizations such as SPEED etc., larger num-samples obtain better efficiency on all layers while

*Table 1.* Accuracy and cross entropy (CE) loss of Tandem Transformers with respect to ground truth labels as well as the predictions of the primary model $\mathcal{M}_L$, PaLM2-Bison. As is clear from the results, the Tandem model of PaLM2-Gecko and PaLM2-Bison substantially outperforms the stand alone PaLM2-Gecko model.

| | PaLM2-Gecko | PaLM2-Gecko-Distil | Tandem-CE **(ours)** | Tandem-Distil **(ours)** |
|---|---|---|---|---|
| Accuracy (GT) | 55.06 | 56.50 | 58.35 | **58.61** |
| CE loss (GT) | 2.14 | 2.12 | **1.94** | 1.99 |
| Relative accuracy | 74.64 | 75.30 | 80.00 | **81.00** |
| Relative TV distance | 0.391 | 0.318 | 0.178 | **0.141** |

**Adaptive Block Length.** While standard SPEED uses a fixed block length $\gamma$, we introduce an adaptive approach. We train a relatively small 2-layer multi-layer perceptron – *router MLP* – model to predict whether the current draft token from $\mathcal{M}_S$ is likely to be accepted by the primary model $\mathcal{M}_L$. At each timestep, we compare the prediction of this small model to a threshold $\tau$, deciding whether to: a. Verify with $\mathcal{M}_L$, or b. Continue drafting with $\mathcal{M}_S$.

Input features to the router MLP are: $\mathcal{M}_S$'s entropy over the current token's vocabulary distribution, top-$k$ probabilities for the current token for an appropriate $k$, and $\mathcal{M}_S$'s model embeddings corresponding to these top-$k$ most probable tokens. We train the router MLP to predict the probability of disagreement using cross-entropy loss, with ground truth being: $TV(y_j^S, y_j^P)$, where $TV(y_j^S, y_j^P)$ is the total variation (TV) distance between the output logits of $\mathcal{M}_S$ and $\mathcal{M}_L$ for the $j^{\text{th}}$ token.

## 4. Experiments

In this section, we present experimental results evaluating Tandem Transformer models. Except for the new architecture of Tandem Transformers, we generally follow the same training protocols as described in (Anil et al., 2023), including the training dataset, optimizer, etc.

**Further Training Details.** For both Tandem-CE and Tandem-Distil, we initialize the secondary model $\mathcal{M}_S$ to be the pretrained PaLM2-Gecko, while freezing primary model $\mathcal{M}_L$ to be the pretrained PaLM2-Bison (Anil et al., 2023). The projection/Tandem feedforward layers are chosen to be linear layers and initialized randomly. Both the Tandem models – Tandem-CE and Tandem-Distil– are trained with a block length of $\gamma = 2$. For our evaluation within the SPEED framework, we consider a logit distillation version of PaLM2-Gecko, called PaLM2-Gecko-Distil, which is initialized with the PaLM2-Gecko model and then trained using logit distillation, similar to the second phase of training of the Tandem-Distil model, since distillation has been shown to help improve the secondary models in SPEED (Zhou et al., 2023).

larger batch size obtains better efficiency only on feedforward and softmax layers, and not the attention layer.

**Adaptive Block Length in SPEED.** We train a small, 2-layer MLP model to predict whether the current drafter token from $\mathcal{M}_S$ is likely to be accepted by primary model $\mathcal{M}_L$. We set $\tau = 0.8$ as the threshold to determine if $\mathcal{M}_S$ can continue generating more tokens.

### 4.1. Performance Evaluation

We compare the performance of Tandem-CE and Tandem-Distil against PaLM2-Gecko, PaLM2-Gecko-Distil, PaLM2-Otter and PaLM2-Bison on several downstream tasks.

For downstream task evaluation, we compare on SuperGLUE (Wang et al., 2019), TydiQA (Clark et al., 2020), a large collection of generation tasks, which we call Gen-tasks (comprising of SQuADv2 (Rajpurkar et al., 2018), Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), WebQuestions (Berant et al., 2013) and Lambada (Paperno et al., 2016)), MBPP (Austin et al., 2021), and WMT22 (Zerva et al., 2022). WMT22 results are averaged over x $\rightarrow$ en translations for different languages x. For TydiQA, we pass the gold passage as part of the input, and report the average F1-score over all languages. For SuperGLUE and Gen-tasks, we follow the experimental settings as described in (Anil et al., 2023) and report the average results. We report 1-shot evaluations for all performance evaluation experiments.

### 4.2. Latency Evaluation

We perform latency evaluation in two different settings. In the first setting, we use Tandem-Distil as the secondary model within SPEED, with PaLM2-Bison as the primary model. Note that the SPEED framework guarantees that the outputs will be of the same quality as the primary model PaLM2-Bison. For comparison, we use PaLM2-Bison as a stand alone model, as well as SPEED with PaLM2-Bison as the primary model and PaLM2-Gecko-Distil as the secondary model as our baselines. In the second setting, we evaluate the latency of Tandem-CE and Tandem-Distil as stand alone models with PaLM2-Gecko, PaLM2-Otter and PaLM2-Bison. All the evaluations are performed on TPUv5e (Cloud).

We evaluate latency on the test sets of CNN/DailyMail (Hermann et al., 2015), Reddit Posts summarization (Kim et al., 2018), and 1000 prompts from the 1 Billion Word Benchmark (Chelba et al., 2014). We report latency results for both num-samples $= 1$ as well as $4$.

*Table 2.* End-to-end latency gain of various secondary models, when used within the SPEED framework with PaLM2-Bison as the primary model. The secondary models we consider are: PaLM2-Gecko-Distil and Tandem-Distil. Since Tandem-Distil has better acceptance rate compared to PaLM2-Gecko-Distil, e.g., for $\gamma = 5$, Tandem-Distil has, on average, $11.24\%$ more tokens accepted compared to PaLM2-Gecko-Distil, for each secondary model, and on each dataset, we use the optimal block length $\gamma$ parameter. We consider two settings, one where we generate a single response and another where we generate 4 responses for the given query. The third and fourth column provide the speedup by using PaLM2-Gecko-Distil and Tandem models respectively, with respect to the PaLM2-Bison model. The last column indicates the relative gain of using the Tandem model as the secondary model in SPEED, instead of PaLM2-Gecko-Distil. The results clearly demonstrate the additional improvements Tandem obtains, on top of logit distillation.

| Dataset | num-samples | PaLM2-Gecko-Distil (baseline) | Tandem-Distil (**ours**) | Tandem-Distil (**ours**; relative gain) |
|---|---|---|---|---|
| Reddit | 1 | $2.169\times$ $(\gamma = 7)$ | $2.471\times$ $(\gamma = 7)$ | $\mathbf{1.139}\times$ |
| | 4 | $1.919\times$ $(\gamma = 5)$ | $2.234\times$ $(\gamma = 7)$ | $\mathbf{1.164}\times$ |
| CNN/DailyMail | 1 | $2.219\times$ $(\gamma = 7)$ | $2.473\times$ $(\gamma = 7)$ | $\mathbf{1.115}\times$ |
| | 4 | $1.940\times$ $(\gamma = 5)$ | $2.190\times$ $(\gamma = 7)$ | $\mathbf{1.129}\times$ |
| LM1B | 1 | $2.348\times$ $(\gamma = 7)$ | $2.610\times$ $(\gamma = 7)$ | $\mathbf{1.112}\times$ |
| | 4 | $2.011\times$ $(\gamma = 5)$ | $2.359\times$ $(\gamma = 7)$ | $\mathbf{1.173}\times$ |

*Table 3.* Standalone evaluation of the Tandem model (with $\gamma = 3$). The first five rows present downstream evaluations of the Tandem Transformers on a variety of generative and ranking tasks. We see that the Tandem model substantially improves upon the performance of stand alone PaLM2-Gecko model, and is on par with the PaLM2-Otter model. On the other hand, the latency evaluations in the last row demonstrate that the Tandem model is about $1.16\times$ faster than the PaLM2-Otter model.

| Dataset | PaLM2-Gecko | Tandem-CE **(ours)** | Tandem-Distil **(ours)** | PaLM2-Otter | PaLM2-Bison |
|---|---|---|---|---|---|
| Generative-tasks | 28.8 | 37.1 | 44.0 | 51.1 | 57.5 |
| MBPP | 4.8 | 13.8 | 21.2 | 20.8 | 30.4 |
| WMT22-1shot-to-nonenglish | 35.1 | 37.4 | 44.1 | 48.4 | 50.5 |
| TydiQA-GoldP | 55.0 | 65.7 | 69.0 | 69.7 | 73.4 |
| SuperGLUE | 62.8 | 78.5 | 78.8 | 79.0 | 81.5 |
| Speedup over PaLM2-Bison | $6.40\times$ | $2.75\times$ | $2.75\times$ | $2.36\times$ | $1\times$ |

### 4.3. Evaluation Results

We now present results of our evaluation of Tandem Transformers.

**Pretraining Metrics.** Table 1 presents a comparison of accuracy and cross entropy (CE) loss of various baselines as well as Tandem models, with respect to both the ground truth labels as well as the primary model $\mathcal{M}_L$'s predictions. As we can see, Tandem Transformers performs better than logit distillation, while combining logit distillation with Tandem Transformers, further improves its performance.

**Latency within SPEED.** Table 2 presents results on the latency of Tandem Transformers within the SPEED framework. Specifically, we compare the speedup obtained over the PaLM2-Bison model, by using SPEED with PaLM2-Gecko-Distil as the secondary model vs Tandem-Distil as the secondary model. The results clearly demonstrate the improvements obtained by Tandem on top of distillation. Table 7 in Appendix A presents the speedups computed only over the decode time (i.e., excluding the query processing time). Note that since the SPEED framework guarantees that the outputs are of same quality as those of the primary model, PaLM2-Bison, the latency improvements given by the Tandem model do not have any quality tradeoffs.

**Evaluation as a Standalone Model.** We evaluate the Tandem model as a stand alone model in its own right. Table 3 presents a comparison of both downstream evaluations on standard downstream benchmarks, as well as latency evaluations. As can be seen, the Tandem model substantially improves upon the downstream performance of the baseline model, and is almost on par with the PaLM2-Otter model. Detailed results presented in Tables 9 and 10 in Appendix A show that, in some cases, the Tandem model is closer to the PaLM2-Bison model itself. At the same time, the Tandem model is about $1.16\times$ times faster compared to the PaLM2-Otter model, making it a compelling candidate for stand

*Table 4.* End-to-end latency gain of various secondary models, when used within the SPEED framework with PaLM2-Bison as the primary model. The secondary models we consider are: PaLM2-XXXS-Distil and Tandem-Distil (with PaLM2-XXXS as the secondary model). For each secondary model, we tune the optimal block length $\gamma$ parameter. We consider two settings, one where we generate a single response and another where we generate $4$ responses for the given query. The third and fourth column provide the speedup by using PaLM2-XXXS-Distil and Tandem models respectively, with respect to the PaLM2-Bison model. The last column indicates the relative gain of using the Tandem model as the secondary model in SPEED, instead of PaLM2-XXXS-Distil. The results clearly demonstrate the additional improvements Tandem obtains, on top of logit distillation. Note that Tandem-Distil (with PaLM2-XXXS as the secondary model) improves over its PaLM2-Gecko counterpart.

| Dataset | num-samples | PaLM2-XXXS-Distil (baseline) | Tandem-Distil (**ours**) | Tandem-Distil (**ours**; relative gain) |
|---|---|---|---|---|
| LM1B | 1 | $2.445\times$ $(\gamma = 5)$ | $3.040\times$ $(\gamma = 5)$ | $\mathbf{1.243}\times$ |
| | 4 | $1.821\times$ $(\gamma = 5)$ | $2.488\times$ $(\gamma = 5)$ | $\mathbf{1.366}\times$ |

*Table 5.* End-to-end latency speedup obtained by Tandem-Distil + SPEED + Adaptive $\gamma$ on different evaluation datasets. The second and third columns show the speedup over the stand alone PaLM2-Bison model and Tandem-Distil + SPEED model respectively. The latency is evaluated for generating a single response. Adaptive $\gamma$ enables us to use much larger block lengths without losing performance. For example, on the Reddit dataset, the optimal $\gamma$ for the Tandem model in the standard SPEED setup is 7, while adaptive $\gamma$ obtains better results with $\gamma_{\text{max}} = 17$.

| Dataset | PaLM-Bison | Tandem-Distil + SPEED |
|---|---|---|
| Reddit | $2.582\times$ $(\gamma_{\text{max}} = 17)$ | $\mathbf{1.045}\times$ |
| CNN/ DailyMail | $2.599\times$ $(\gamma_{\text{max}} = 17)$ | $\mathbf{1.051}\times$ |
| LM1B | $2.853\times$ $(\gamma_{\text{max}} = 27)$ | $\mathbf{1.093}\times$ |

alone deployment as well.

**Adaptive Block Length.** We now present a way to improve the performance of SPEED with adaptive block lengths (Adaptive $\gamma$ or AG), where after every token predicted by the secondary model, we use a small, inexpensive router to determine whether to continue predicting with the secondary model, or verify the tokens generated so far with the primary model. Table 5 presents the speedup obtained by Tandem-Distil + SPEED + AG compared with the PaLM2-Bison model as well as the Tandem-Distil + SPEED model. Table 8 in Appendix A presents the speedup as measured only over the decode component of the latency i.e., excluding query processing time.

In Table 6, we present the number of primary model, and secondary model runs for Tandem-Distil + SPEED and Tandem-Distil + SPEED + Adaptive $\gamma$. The results put forth the benefits of using an adaptive block length, since it drastically reduces the number of secondary model runs while slightly increasing the number of primary model runs.

**Smaller Secondary Model.** While Tandem-Distil show significant improvements over PaLM2-Bison, we find that using a smaller secondary model for Tandem Transformers further boosts the latency improvements. Table 4 presents results for the same where PaLM2-Gecko is replaced with PaLM2-XXXS, with PaLM2-Bison being retained as the primary model. As can be seen, this setting further improves the latency gains of the Tandem model, whereas PaLM2-XXXS-Distil improves over its PaLM2-Gecko counterpart only for num-samples $= 1$, indicating that Tandem Transformers provide considerably better latency, even with extremely small secondary models.

*Table 6.* Primary model and secondary model runs for Tandem-Distil and Tandem-Distil + AG on the LM1B benchmark. Note that these results are obtained for num-samples $= 1$. We can see that the number of secondary model runs have come down by $90$ whereas the number of large model runs has gone up only by $3$. The results clearly showcase that an adaptive block length can significantly cut down on the number of secondary model runs and give non-trivial latency gains.

| | Tandem-Distil | Tandem-Distil + AG |
|---|---|---|
| Primary model runs | 51.53 | 54.67 |
| Secondary model runs | 360.73 | 271.63 |

## 5. Conclusions and Discussion

In this work, we introduce a novel architecture, Tandem Transformers, which combines a small autoregressive model with a large model operating in block mode. Tandem Transformers substantially boost the small model's predictive accuracy by allowing it to attend to representations from the large model. In our experiments, a Tandem model comprising of PaLM2-Bison and PaLM2-Gecko substantially improves over a standalone PaLM2-Gecko, and gives comparable performance to the PaLM2-Otter model, while being $1.16\times$ faster than the PaLM2-Otter model. When used within the SPEED setup as a secondary model, the distilled *Tandem* PaLM2-Gecko model gives around $1.14\times$ speedup

over a distilled PaLM2-Gecko model. We further improve our Tandem model through an adaptive block length procedure in SPEED and obtain around $1.22\times$ speedup over using PaLM2-Gecko-Distil as the secondary model.

**Limitations and Future directions**

- **Other variants of Tandem**: In our current approach, we use the large model only through its representations of the past tokens. Is it possible to use the large model to also generate a *plan for the future $\gamma$ tokens*?
- **Alternative to LoRA for finetuning**: The current approach for finetuning a base model for multiple downstream applications is through low rank adaptation (LoRA) (Hu et al., 2021). It will be interesting to explore whether Tandem with block length 0 can be an effective alternative to LoRA, while reducing the training cost substantially since backpropagation needs to be done only for the small model.
- **Adaptive $\gamma$ for larger num-samples/batch-size:** While we see promising results with adaptive $\gamma$ in SPEED for num samples 1, extending it to larger num samples seems challenging. Identifying an effective way of determining when to continue generating with small model vs verifying with large model, in the larger num samples setting, is also an interesting direction of future work.

## Impact Statement

Our work provides a more computationally efficient large language model inference solution, which we hope can bring down carbon emissions associated with LLM inference. It also helps with easier deployment of LLMs, which could have potential societal consequences, that seem difficult to predict.

## References


Anil, R. et al. Palm 2 technical report, 2023.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C. J., Terry, M., Le, Q. V., and Sutton, C. Program synthesis with large language models. *ArXiv*, abs/2108.07732, 2021. URL https://api.semanticscholar.org/CorpusID:237142385.

Berant, J., Chou, A., Frostig, R., and Liang, P. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1533–1544. ACL, 2013. URL https://aclanthology.org/D13-1160/.

Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J. D., Chen, D., and Dao, T. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. *CoRR*, abs/2401.10774, 2024. doi: 10.48550/ARXIV.2401.10774. URL https://doi.org/10.48550/arXiv.2401.10774.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., and Robinson, T. One billion word benchmark for measuring progress in statistical language modeling. In Li, H., Meng, H. M., Ma, B., Chng, E., and Xie, L. (eds.), *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pp. 2635–2639. ISCA, 2014. doi: 10.21437/INTERSPEECH.2014-564. URL https://doi.org/10.21437/Interspeech.2014-564.

Clark, J., Choi, E., Collins, M., Garrette, D., Kwiatkowski, T., Nikolaev, V., and Palomaki, J. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470, 2020. URL https://api.semanticscholar.org/CorpusID:212657414.

Cloud, G. Cloud tpu v5e inference. URL https://cloud.google.com/tpu/docs/v5e-inference. Accessed on Feb 1, 2024.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

He, Z., Zhong, Z., Cai, T., Lee, J. D., and He, D. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*, 2023.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pp. 1693–1701, 2015.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In Barzilay, R. and Kan, M. (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pp. 1601–1611. Association for Computational Linguistics, 2017. doi: 10.18653/V1/P17-1147. URL https://doi.org/10.18653/v1/P17-1147.

Kim, B., Kim, H., and Kim, G. Abstractive summarization of reddit posts with multi-level memory networks. *CoRR*, abs/1811.00783, 2018. URL http://arxiv.org/abs/1811.00783.

Kim, S., Mangalam, K., Moon, S., Malik, J., Mahoney, M. W., Gholami, A., and Keutzer, K. Speculative decoding with big little decoder. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A. P., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466, 2019. doi: 10.1162/TACL_A_00276. URL https://doi.org/10.1162/tacl_a_00276.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Li, Z., You, C., Bhojanapalli, S., Li, D., Rawat, A. S., Reddi, S. J., Ye, K., Chern, F., Yu, F., Guo, R., et al. The lazy neuron phenomenon: On emergence of activation sparsity in transformers. In *The Eleventh International Conference on Learning Representations*, 2022.

Liu, Z., Wang, J., Dao, T., Zhou, T., Yuan, B., Song, Z., Shrivastava, A., Zhang, C., Tian, Y., Re, C., et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, pp. 22137–22176. PMLR, 2023.

Mudgal, S., Lee, J., Ganapathy, H., Li, Y., Wang, T., Huang, Y., Chen, Z., Cheng, H., Collins, M., Strohman, T., Chen, J., Beutel, A., and Beirami, A. Controlled decoding from language models. *CoRR*, abs/2310.17022, 2023. doi: 10.48550/ARXIV.2310.17022. URL https://doi.org/10.48550/arXiv.2310.17022.

Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernández, R. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/V1/P16-1144. URL https://doi.org/10.18653/v1/p16-1144.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don't know: Unanswerable questions for squad. In Gurevych, I. and Miyao, Y. (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pp. 784–789. Association for Computational Linguistics, 2018. doi: 10.18653/V1/P18-2124. URL https://aclanthology.org/P18-2124/.

Stern, M., Shazeer, N., and Uszkoreit, J. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Zerva, C., Blain, F., Rei, R., Lertvittayakumjorn, P., De Souza, J. G., Eger, S., Kanojia, D., Alves, D., Orăsan, C., Fomicheva, M., et al. Findings of the wmt 2022 shared task on quality estimation. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pp. 69–99, 2022.

Zhou, Y., Lyu, K., Rawat, A. S., Menon, A. K., Rostamizadeh, A., Kumar, S., Kagy, J.-F., and Agarwal, R. Distillspec: Improving speculative decoding via knowledge distillation. *arXiv preprint arXiv:2310.08461*, 2023.

# A. Additional Results

In this section, we will present additional experimental results.

## A.1. Decode Time Results

In Tables 7 and 8, we compare the decode time results (i.e. end-to-end time − time required to process the input prefix) of our Tandem model and its Adaptive $\gamma$ variant with the baselines.

*Table 7.* Decode-time-only latency gain of various secondary models, when used within the SPEED framework with PaLM2-Bison as the primary model. The secondary models we consider are: PaLM2-Gecko-Distil and Tandem-Distil. For each secondary model, and on each dataset, we use the optimal block length $\gamma$ parameter. We consider two settings, one where we generate a single response and another where we generate 4 responses for the given query. The third and fourth column provide the speedup by using PaLM2-Gecko-Distil and Tandem models respectively, with respect to the PaLM2-Bison model. The last column indicates the relative gain of using the Tandem model as the secondary model in SPEED, instead of PaLM2-Gecko-Distil. The results clearly demonstrate the additional improvements Tandem obtains, on top of logit distillation.

| Dataset | num-samples | PaLM2-Gecko-Distil (baseline) | Tandem-Distil (**ours**) | Tandem-Distil (**ours**; relative gain) |
|---|---|---|---|---|
| Reddit | 1 | $2.356\times$ $(\gamma = 7)$ | $2.737\times$ $(\gamma = 7)$ | $\mathbf{1.162}\times$ |
| | 4 | $2.042\times$ $(\gamma = 5)$ | $2.425\times$ $(\gamma = 7)$ | $\mathbf{1.188}\times$ |
| CNN/DailyMail | 1 | $2.418\times$ $(\gamma = 7)$ | $2.740\times$ $(\gamma = 7)$ | $\mathbf{1.133}\times$ |
| | 4 | $2.066\times$ $(\gamma = 5)$ | $2.369\times$ $(\gamma = 7)$ | $\mathbf{1.146}\times$ |
| LM1B | 1 | $2.460\times$ $(\gamma = 7)$ | $2.756\times$ $(\gamma = 7)$ | $\mathbf{1.120}\times$ |
| | 4 | $2.080\times$ $(\gamma = 5)$ | $2.466\times$ $(\gamma = 7)$ | $\mathbf{1.186}\times$ |

*Table 8.* Decode-time-only latency speedup obtained by Tandem-Distil + SPEED + Adaptive $\gamma$ on different evaluation datasets. The second and third columns show the speedup over the stand alone PaLM2-Bison model and Tandem-Distil + SPEED model respectively. The latency is evaluated for generating a single response. Adaptive $\gamma$ enables us to use much larger block lengths without losing performance. For example, on the Reddit dataset, the optimal $\gamma$ for the Tandem model in the standard SPEED setup is 7, while adaptive $\gamma$ obtains better results with $\gamma_{\text{max}} = 17$.

| Dataset | PaLM-Bison | Tandem-Distil + SPEED |
|---|---|---|
| Reddit | $2.885\times$ $(\gamma_{\text{max}} = 17)$ | $\mathbf{1.054}\times$ |
| CNN/DailyMail | $2.908\times$ $(\gamma_{\text{max}} = 17)$ | $\mathbf{1.061}\times$ |
| LM1B | $3.040\times$ $(\gamma_{\text{max}} = 27)$ | $\mathbf{1.103}\times$ |

## A.2. Detailed Performance Evaluation Results

In Table 9, we present results for our Tandem model and the compared baselines on each individual task in Generative-tasks. Likewise, in Table 10 we present results on each individual task in SuperGLUE.

# B. Inference of Tandem Transformers

Figure 3 presents the inference for Tandem Transformers without the the free token from the primary model $\mathcal{M}_L$.

*Table 9.* Evaluation of the Tandem model on each of the Generative-tasks. We see that the Tandem model substantially improves upon the performance of stand alone PaLM2-Gecko model, and on most datasets, is on par with the PaLM2-Otter model.

| Dataset | PaLM2-Gecko | Tandem-CE **(ours)** | Tandem-Distil **(ours)** | PaLM2-Otter | PaLM2-Bison |
|---|---|---|---|---|---|
| Lambada (acc = Accuracy) | 45.5 | 59.2 | 68.3 | 78.9 | 82.9 |
| NaturalQuestions (em = Exact Match) | 7.7 | 9.9 | 14.4 | 19.9 | 28.1 |
| SQuADv2 (em) | 45.3 | 67.8 | 70.2 | 70.3 | 75.4 |
| TriviaQA (em) | 36.8 | 36.9 | 51.2 | 68.9 | 77.3 |
| WebQuestions (em) | 9.0 | 12.0 | 16.0 | 17.6 | 23.8 |

*Table 10.* Evaluation of the Tandem model on each of the SuperGLUE tasks. We see that the Tandem model substantially improves upon the performance of stand alone PaLM2-Gecko model, and on most datasets, is on par with the PaLM2-Otter model.

| Dataset | PaLM2-Gecko | Tandem-CE **(ours)** | Tandem-Distil **(ours)** | PaLM2-Otter | PaLM2-Bison |
|---|---|---|---|---|---|
| BoolQ (acc) | 65.4 | 87.8 | 87.6 | 85.5 | 88.8 |
| CB (acc) | 39.3 | 82.1 | 83.9 | 71.4 | 87.5 |
| COPA (acc) | 80.0 | 78.0 | 82.0 | 88.0 | 88.0 |
| RTE (acc) | 55.2 | 80.1 | 78.3 | 84.1 | 77.6 |
| ReCoRD (acc) | 85.5 | 87.8 | 87.2 | 91.2 | 92.2 |
| WIC (acc) | 47.5 | 50.0 | 50.6 | 49.7 | 50.9 |
| WSC (acc) | 75.8 | 81.1 | 80.4 | 86.3 | 86.3 |
| MultiRC (F1) | 53.9 | 80.8 | 80.1 | 76.1 | 80.5 |

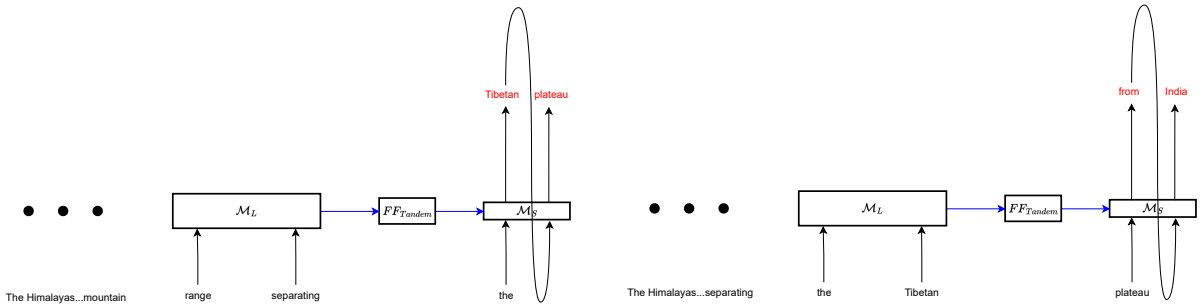


*Figure 3.* Inference of Tandem Transformers *without* free token from the primary model $\mathcal{M}_L$. (left) First block prediction. (right) Second block prediction. Given the same query *The Himalayas are a mountain range separating the* as in Figure 2, here, $\mathcal{M}_L$ first processes this query except the last token *the*. The last token is passed as an input to the secondary model $\mathcal{M}_S$, which attends to $\mathcal{M}_L$ representations for all past tokens, and produces the first block of responses Tibetan plateau autoregressively. In the second block, $\mathcal{M}_L$ processes *the Tibetan* in a block mode while *plateau* is passed as an input to $\mathcal{M}_S$, which then autoregressively generate the next block of response from India. This eventually leads to a response of Tibetan plateau from India....