U-ResBM: Rank-Decreasing Skip Connections for Communication-Efficient Pipeline Parallelism

Marian-Sergiu Nistor
Independent Researcher
sergiunistor.info@gmail.com

Abstract

Decentralized LLM training relies on pipeline parallelism, splitting a model into stages spread across machines connected over the internet, with each stage passing activations forward and gradients back on every step. Over an internet connection these transfers dominate the time per step, and at full size they are too large to send, in most cases. Residual Bottleneck Models (ResBM) reduce that size by routing each pipeline boundary through a learned encoder-decoder bottleneck while keeping a low-rank identity path intact, which buys aggressive compression without slowing convergence. However, each boundary passes only a narrow, low-rank activation, so high-rank information and gradients cannot cross it. We introduce **U-ResBM**, which adds long-range residual connections that carry higher-rank information across directly. Each connection links an early layer to its mirror in the second half, and its rank grows linearly with the number of stages it spans, so the outermost connection carries the highest-rank information and the rank decreases toward the middle. Because a connection's rank scales with how many stages it crosses, sending it takes exactly the time the pipeline needs to reach the target stage, so the data arrives just in time. These connections add no parameters and leave the per-boundary compression unchanged. Trained from scratch on C4 at 128x compression, **U-ResBM** converges faster than ResBM and to a lower final loss under both AdamW and Muon, achieving the best results under Muon.

1 Introduction

Pipeline parallelism trains a model whose parameters exceed the memory of a single accelerator by dividing its layers into consecutive stages and then placing each stage on a separate device [Huang et al., 2019, Narayanan et al., 2021]. Because the stages must exchange activations and gradients on every step, this depends on the fast, high-bandwidth interconnects found inside a datacenter.

Such high-throughput interconnects are confined to dedicated clusters. However, a growing line of work trains models on consumer-level machines that are linked through the internet, which would make training large models possible without access to datacenter-grade hardware [Diskin et al., 2021, Yuan et al., 2022, Ryabinin et al., 2023]. For data parallelism, which places a full replica of the model on every device and splits the training data among them, a worker can synchronize with the others infrequently [Douillard et al., 2023] or send a compressed form of its gradients in place of the full tensors [Peng et al., 2026], which reduces the volume communicated between workers by multiple orders of magnitude. Nevertheless, pipeline parallelism resists the same treatment, because it has to send the activations and their gradients between stages on every single step, and the compression schemes designed for data-parallel gradients do not work on those activations, since only a limited amount of compression is possible before the model fails to match the quality of an uncompressed run [Bian et al., 2024, Rudakov et al., 2023, Ramasinghe et al., 2026, Aboudib et al., 2026]. Once the compression is pushed past that point, the model no longer converges [Ramasinghe et al., 2026],

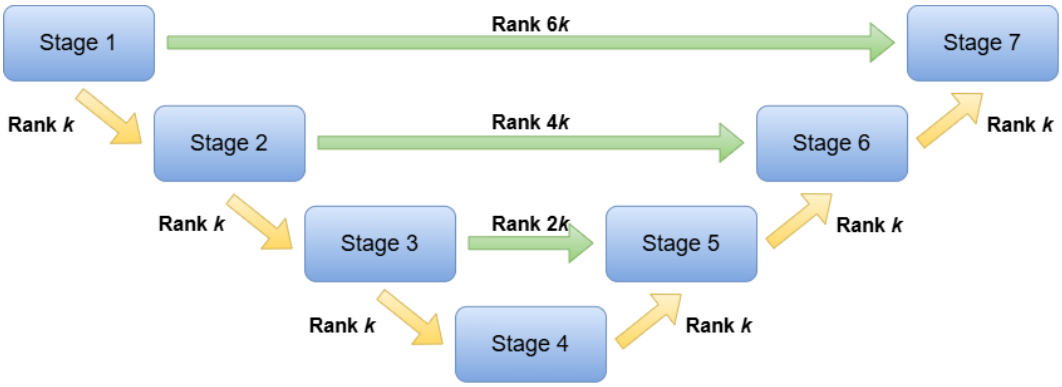


Figure 1: **U-ResBM** across seven pipeline stages. Each stage passes an ordinary boundary transfer of rank k to the next stage, and mirror-paired stages are joined by a long-range residual whose rank is k times the number of stages it spans, so stage 1 reaches stage 7 at rank $6k$, stage 2 reaches stage 6 at rank $4k$, and stage 3 reaches stage 5 at rank $2k$. The rank of the residuals decreases toward the center.

which requires that the compression method preserves the residual path for stable gradient flow [He et al., 2016].

Two recent methods take on this problem directly. Subspace Networks [Ramasinghe et al., 2026] fix a shared low-rank subspace and constrain every layer's activations and gradients to lie within it, so that only the low-dimensional components need to be transmitted and the receiving device can reconstruct the full activation from them exactly, which reaches up to 100x compression while still converging as well as an uncompressed model. ResBM [Aboudib et al., 2026] instead places a small learned bottleneck at each boundary between stages, shrinking the activation before sending it, and then expanding it again on the receiving end, while a low-rank copy of the identity path is carried alongside the bottleneck. The encoder and decoder weights are learned together with the rest of the model by ordinary gradient descent rather than through a separate optimization procedure.

ResBM confines every transfer between stages to the same k dimensions, yet the representation each stage produces before that bottleneck is not itself low-rank. Measuring the effective rank of the model's weights over the course of training, and the effective rank of the activations and gradients at the end of the training run, we find that it stays high throughout rather than collapsing, which shows that the model keeps driving far more than k dimensions of structure into a channel that can carry only k, discarding a lot of information.

We introduce **U-ResBM**, which keeps this per-boundary bottleneck but restores the lost rank through a set of long-range residual connections that run directly between distant stages. Following the long skip connections that join the contracting and expanding halves of a U-Net [Ronneberger et al., 2015], each connection links an early stage to its mirror in the second half of the network and carries a rank equal to k times the number of stages it spans, so the outermost connection carries the highest-rank information and the rank decreases toward the middle. Because that rank grows linearly with the span, the transfer of each connection takes exactly as long as the pipeline needs to reach its target stage, so its data arrives just in time, without increasing latency. These connections introduce no parameters and leave the per-boundary compression unchanged.

Our contributions are the following:

- We show through an effective-rank analysis that the bottleneck operates at high rank, so the model drives more than k dimensions of structure through a channel that can carry only k.
- We introduce **U-ResBM**, a set of long-range, rank-decreasing residual connections added on top of ResBM that restore higher-rank information flow between distant stages without introducing parameters or changing the per-boundary compression.
- We pretrain a model from scratch at $128\times$ compression and show that **U-ResBM** converges faster and to a lower final loss than ResBM under both AdamW and Muon, reaching its best results under Muon.

- We show that setting each connection's rank to k times the number of stages it spans makes its network transfer finish exactly when the pipeline reaches the target stage, so the added communication overlaps with computation and costs no extra latency.

2 Related Work

Subspace Networks [Ramasinghe et al., 2026] predefine a low-dimensional subspace and keep the weights in the subspace, so that each activation, and the gradient that flows back through it, splits into a low-rank part that varies within the subspace and a high-rank part that the receiving stage can already reproduce. Only the varying part is sent, and the full tensor is rebuilt exactly at the next layer, so both the forward and backward transfers are compressed. This reaches up to 100x compression with no loss in convergence and has been shown to train billion-parameter models over consumer internet links. The constraint is not enforced in the model architecture itself, but in the optimizer, which is a modified AdamW that holds the projection matrices in the subspace.

ResBM [Aboudib et al., 2026] takes an architectural route instead. It inserts a learned bottleneck encoder and decoder at every block boundary, so that only a k-dimensional activation crosses between stages, and it places an identity shortcut alongside that bottleneck that carries the first k coordinates of the stream through, so the identity path that keeps gradients stable is never cut. Because the compression happens next to the residual stream and not inside it, every weight, the encoder and decoder included, is trained together by an ordinary optimizer with no manifold constraint or separate update. ResBM reports matching the convergence of an uncompressed baseline at 128x compression with a few percent of added parameters. In both Subspace Networks and ResBM, however, every boundary is compressed to the same fixed width, so all information that moves between stages, in either direction, is forced through a single narrow channel.

3 Method

3.1 Residual Bottleneck Models

U-ResBM builds on ResBM [Aboudib et al., 2026], a network of L stages with hidden size H that compresses the information that crosses each stage boundary. At the boundary after stage l, a learned encoder E_l compresses the stage's feed-forward output to a k-dimensional vector c_l with $k \ll H$, and only c_l crosses to the next device, where a decoder D_l expands it back to width H. Beside this bottleneck runs an identity path that passes the first k coordinates of the stream through.

3.2 Rank-Decreasing Residual Connections

U-ResBM adds a set of long-range residuals between mirrored stages, following the skip connections that join the contracting and expanding halves of a U-Net [Ronneberger et al., 2015]. We pair each stage i in the first half of the network with its mirror $j = L - 1 - i$ in the second half, so stage 0 is joined to stage $L - 1$, stage 1 to stage $L - 2$, and so on. A pair is joined only when its two stages are at least two apart, since neighbouring stages already share the ordinary k-dimensional boundary. Each pair carries a residual whose width is set by the distance it spans,

$$r_{ij} = \min\big((j - i)\, k,\ H\big), \tag{1}$$

so the outermost pair carries the highest-rank residual and the width falls as the pairs move toward the middle, which is the U-shape the method is named after. The residual holds no weights of its own, being a slice followed by a zero-pad. We write $[x]_{:r}$ for the first r coordinates of a vector x, and $\mathrm{pad}(x)$ for the vector returned to width H by appending zeros. At the sending stage i we take the first r_{ij} coordinates of the feed-forward output f_i, before the encoder compresses it,

$$u_i = [f_i]_{:r_{ij}}, \tag{2}$$

and at the receiving stage j we pad that slice back to width H and add it onto the decoder output, just before that stage's attention.

At the receiving end, the padded residual is added to the reconstructed input $D_{l-1}(c_{l-1})$ before that stage's attention,

$$d_l = D_{l-1}(c_{l-1}) + \mathrm{pad}(u_{L-1-l}). \tag{3}$$

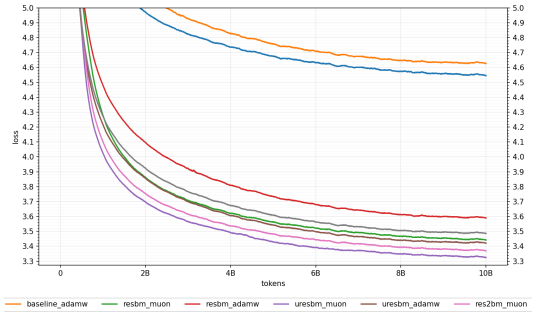


Figure 2: Training loss over 10B tokens for all four architectures under both optimizers. **U-ResBM** with Muon reaches the lowest loss of every run, and under each optimizer **U-ResBM** converges faster and to a lower final loss than ResBM.

The width schedule in Equation 1 is what makes these paths free in terms of execution time as well as in parameters count. A residual that spans $s = j - i$ stages carries $s\,k$ coordinates, and it has to travel from stage i to stage j, which the pipeline reaches only after the forward pass has advanced through those same s stages. Because the amount sent grows in step with the number of stages crossed, its transfer occupies exactly the interval in which the forward pass moves from source to destination, so it can be streamed alongside computation and arrives as the receiving stage begins, without adding a stall. Per stage crossed, a residual therefore moves the same k coordinates as the ordinary boundary already does. Also, the paths add no extra parameters.

4 Experimental Setup

We evaluate **U-ResBM** by pretraining language models from scratch and comparing their convergence against ResBM, an uncompressed baseline, and an ablation of **U-ResBM** that keeps only its longest residual, which we call Res^{2}BM. All models share the Qwen2.5-0.5B architecture [Qwen et al., 2025], a 24-stage transformer with hidden size $H = 896$. The training data used is the English split of the C4 dataset [Raffel et al., 2020], streamed and packed into sequences of length 1024, and each run consumes a Chinchilla-optimal budget of to 10 billion tokens [Hoffmann et al., 2022]. Every configuration uses a batch size of 576 sequences per step.

For the compressed models, the bottleneck dimension is $k = 32$, which sends 32 of the 896 hidden dimensions across each stage boundary, a compression of $128\times$. The encoder and decoder at each boundary are two-layer networks with a hidden width equal to H. The **U-ResBM** residuals follow the schedule of Equation 1, add no parameters, and leave this $128\times$ boundary compression unchanged. Res^{2}BM keeps only the longest of these residuals, the single one joining the first stage to the last.

We train every configuration under two optimizers, AdamW [Loshchilov and Hutter, 2017] and Muon [Liu et al., 2025], so that the effect of the architecture can be read separately from the effect of the optimizer. Both optimizers use a peak learning rate of 2×10^{-4} reached over 500 warmup steps and then decayed by a cosine schedule, to a final 1% of the peak for AdamW and 10% for Muon, with weight decay 0.01 and gradient clipping at 1.0. Training runs in bfloat16 with fp32 master weights and optimizer state.

5 Results

5.1 Convergence

Figure 2 reports training loss for all four architectures under both optimizers. The ordering is the same under either optimizer: **U-ResBM** reaches the lowest loss, the longest-residual ablation Res^{2}BM follows close behind, ResBM is next, and the uncompressed baseline sits well above all three. The Muon variant of **U-ResBM** gives the lowest loss of any run, at roughly 3.32, against roughly 3.44 for ResBM under the same optimizer. Beyond the final value, **U-ResBM** descends faster than ResBM from early in training, so the high-rank residuals help throughout. That Res^{2}BM already recovers

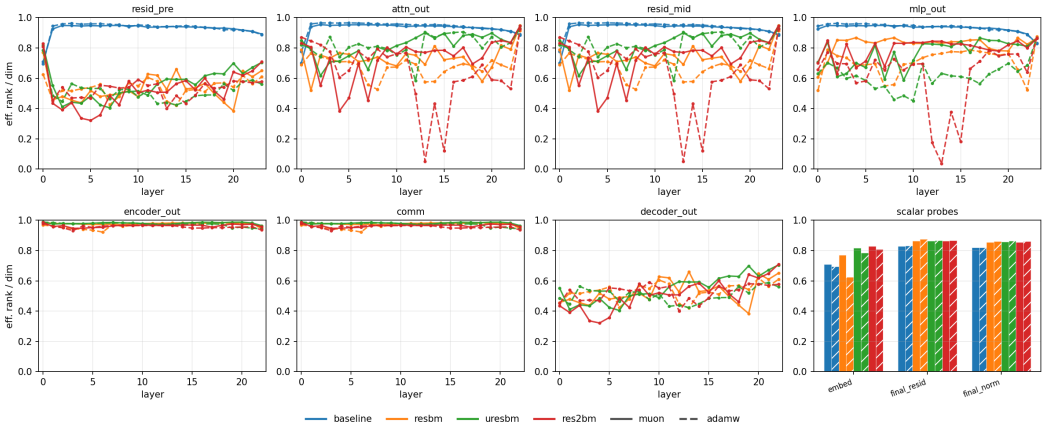


Figure 3: Effective rank of the activation gradients by depth, measured at the end of training. The uncompressed baseline holds near 0.9 across all depth, while every bottleneck model is held well below it, since the backward signal crossing each boundary is squeezed to $k = 32$ dimensions. The communication probe sits near full rank at k, indicating that the channel is used at capacity.

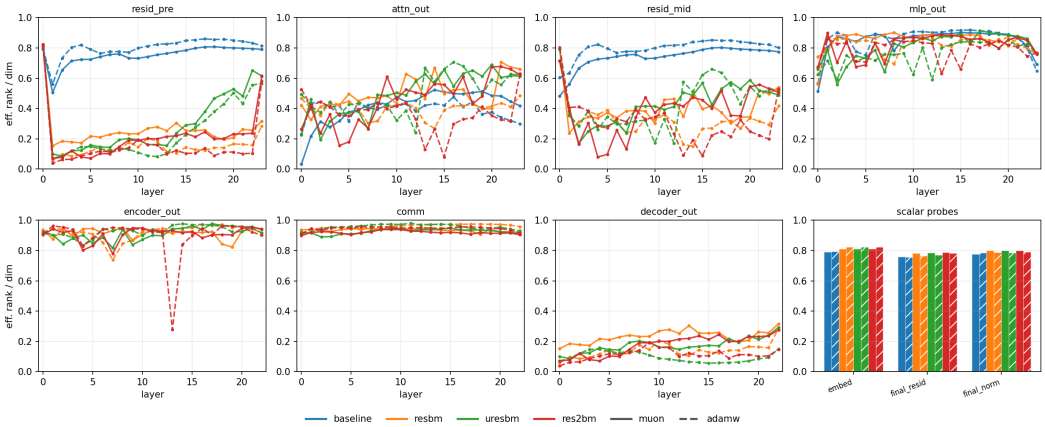


Figure 4: Effective rank of the activations by depth, measured at the end of training. The feed-forward output that the bottleneck compresses (mlp_out) stays high-rank, while the transmitted comm activation is capped at $k = 32$. The forward transfer therefore discards most of the structure the model produces.

most of the distance to **U-ResBM** shows that the single longest residual carries most of the benefit, and that the full rank-decreasing schedule supplies the remainder.

5.2 Activation and Gradient Effective Rank Analysis

Figures 3 and 4 measure the effective rank of the activations and their gradients by depth at the end of training. The feed-forward output that each boundary compresses stays high-rank across the network, and the gradient of that same output is high-rank as well, while the transmitted tensor sits near the full rank of its 32 dimensions. Each boundary therefore takes a representation whose effective rank is high and sends 32, and performs the same down-sizing operation to the gradient on the way back.

At that same feed-forward output, the uncompressed baseline holds its activation-gradient rank near 0.9 at every depth, whereas the bottleneck models are pulled well below that value. This is a direct measurement of what motivates **U-ResBM**, that a single narrow transfer discards most of the structure the model carries, in both the forward and the backward pass, thus providing a motivation for the high-rank residuals.

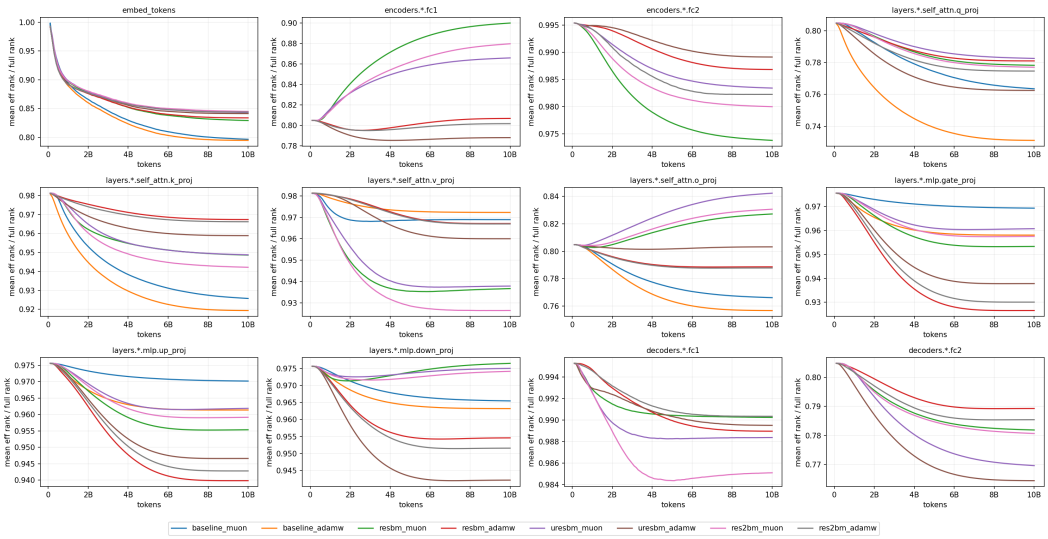


Figure 5: Layer-averaged effective rank of the weights over training. Weight effective rank stays high throughout for every configuration rather than collapsing, and Muon holds it higher than AdamW across the attention and MLP projections.

5.3 Weight Effective Rank Analysis

Figure 5 tracks the layer-averaged effective rank of the weights across training. For every configuration the weights stay high-rank throughout rather than collapsing to a low-dimensional subspace, and across the attention and MLP projections Muon holds this rank higher than AdamW. The Muon runs are also the ones that converge best in Figure 2, so the higher weight rank tracks the stronger convergence, consistent with Muon's rank preservation characteristic.

6 Conclusion

Through an effective-rank analysis of residual bottleneck models, we showed that the compressed activations and their gradients operate at high rank throughout the network, so the model pushes more than k dimensions of structure through a channel that carries only k. We introduced **U-ResBM**, which adds long-range, rank-decreasing skip connections on top of the original ResBM architecture, which link distant stages without adding parameters or changing the per-boundary compression. Pretraining from scratch at $128\times$ compression, **U-ResBM** converges faster and to a lower final loss than ResBM under both AdamW and Muon, and reaches its best result under Muon. Because each connection's rank is set to k times the number of stages it spans, its transfer finishes just as the pipeline reaches the target stage, so the added communication overlaps computation and costs no extra latency.

Limitations

Our experiments consider a single model, Qwen2.5-0.5B, trained on the C4 English split for a budget of 10B tokens. Thus, we do not show how **U-ResBM** behaves at larger scale, on other architectures, or across data distributions. Additionally, the just-in-time property is established from the rank schedule rather than measured as wall-clock latency in a decentralized setting, and confirming that the overlap holds in the context of a real deployment is left to future work.

References

Alan Aboudib, Kalei Brady, Steffen Cruz, et al. Resbm: Residual bottleneck models for low-bandwidth pipeline parallelism. *arXiv preprint arXiv:2604.11947*, 2026.

Song Bian, Dacheng Li, Hongyi Wang, Eric P Xing, and Shivaram Venkataraman. Does compressing activations help model parallel training? *Proceedings of Machine Learning and Systems*, 6: 239–252, 2024.

Michael Diskin, Alexey Bukhtiyarov, Max Ryabinin, Lucile Saulnier, Anton Sinitsin, Dmitry Popov, Dmitry V Pyrkin, Maxim Kashirin, Alexander Borzunov, Albert Villanova del Moral, et al. Distributed deep learning in open collaborations. *Advances in Neural Information Processing Systems*, 34:7879–7897, 2021.

Arthur Douillard, Qixuan Feng, Andrei A Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, and zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384421. doi: 10.1145/3458817.3476209.

Bowen Peng, Lizhang Chen, Baiyu Su, Jeffrey Quesnelle, Diederik Durk Kingma, and Qiang Liu. Decoupled momentum optimization. In *International Conference on Learning Representations*, volume 2026, pages 14058–14083, 2026.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Sameera Ramasinghe, Thalaiyasingam Ajanthan, Gil Avraham, Yan Zuo, and Alexander Long. Subspace networks: Scaling decentralized training with communication-efficient model parallelism. *Advances in Neural Information Processing Systems*, 38:175147–175166, 2026.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Mikhail I Rudakov, Aleksandr Nikolaevich Beznosikov, Ya A Kholodov, and Alexander Vladimirovich Gasnikov. Activations and gradients compression for model-parallel training. In *Doklady Mathematics*, volume 108, pages S272–S281. Springer, 2023.

Max Ryabinin, Tim Dettmers, Michael Diskin, and Alexander Borzunov. Swarm parallelism: Training large models can be surprisingly communication-efficient. In *International Conference on Machine Learning*, pages 29416–29440. PMLR, 2023.

Binhang Yuan, Yongjun He, Jared Davis, Tianyi Zhang, Tri Dao, Beidi Chen, Percy S Liang, Christopher Re, and Ce Zhang. Decentralized training of foundation models in heterogeneous environments. *Advances in Neural Information Processing Systems*, 35:25464–25477, 2022.

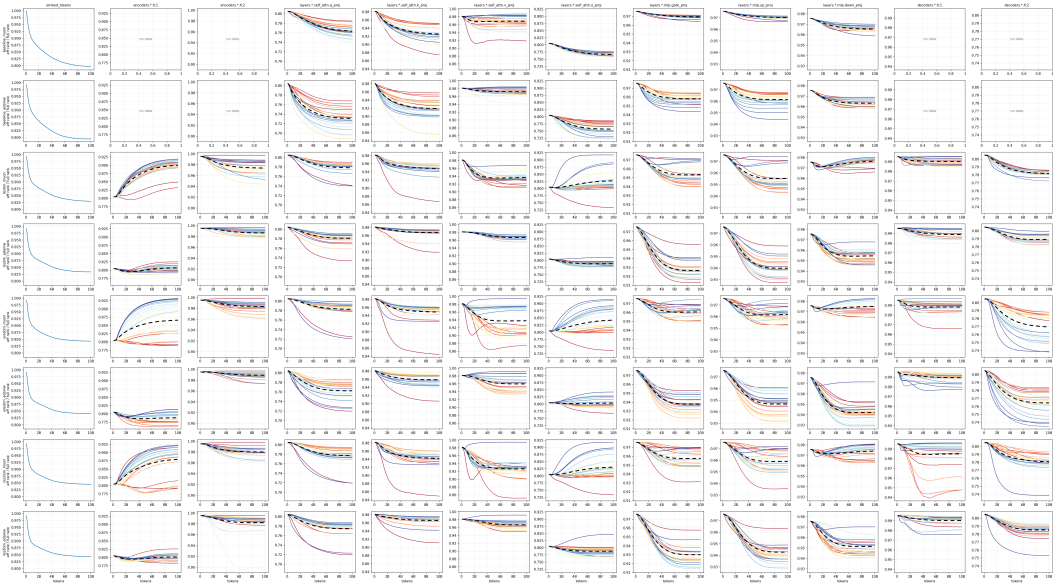


Figure 6: Effective rank of the weights over training for every run and component, with all layers overlaid and shaded from shallow (red) to deep (blue).

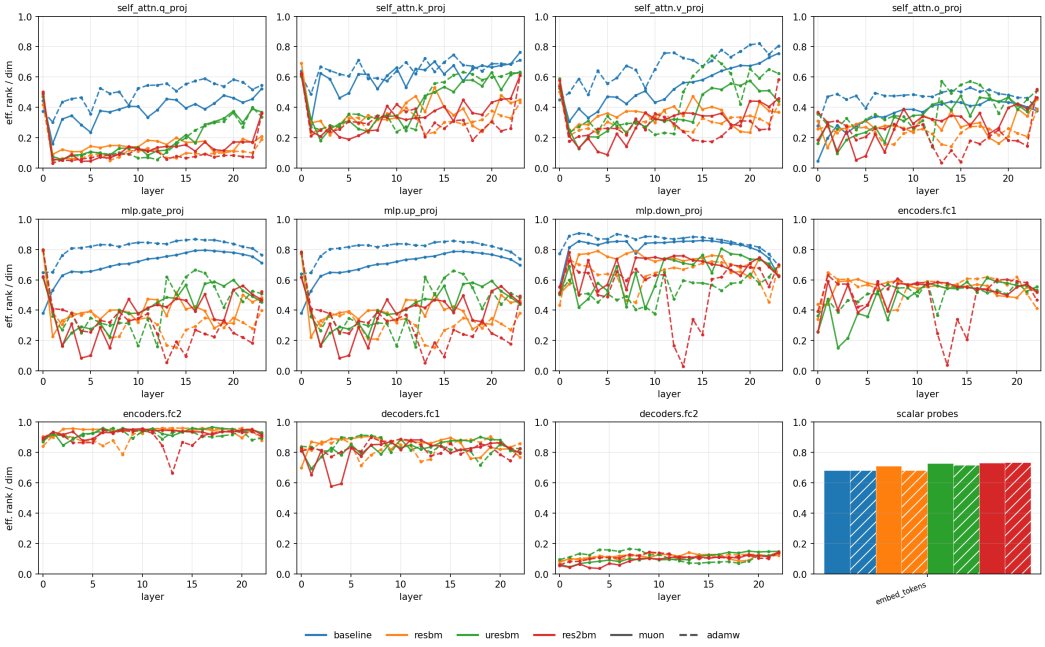


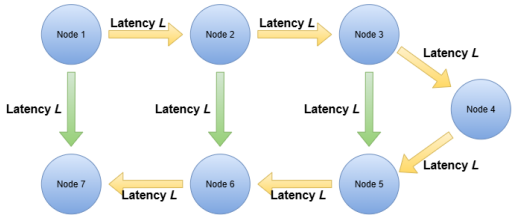
Figure 7: Effective rank of the weight gradients by depth.

A Additional Effective Rank Analysis

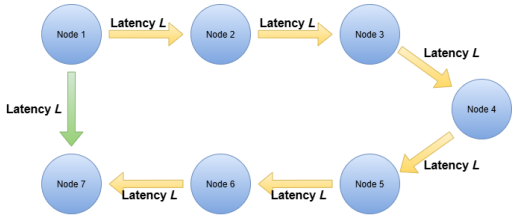
Figure 6 shows the weight effective rank by depth for every run and component. No component in any run collapses to a low-rank subspace: the per-layer bands stay high throughout training, and where they spread they do so by a few percent rather than toward zero. Muon keeps the bands higher than AdamW across the attention and MLP projections, and the bottleneck encoder under Muon gains rank over training at nearly every depth.

Figure 7 reports the effective rank of the weight gradients by depth. The rank varies more by component than by optimizer. The query, gate, and up projections show the widest gap between

(a) ResBM



(b) U-ResBM



(c) Res^{2}BM

Figure 8: The communication topology each architecture requires, considering seven stages under zero jitter. Every edge carries the same latency L: because a residual's rank grows in proportion to the number of stages it spans, the long links move proportionally more data but are issued proportionally earlier, so in this ideal case each edge completes within the same latency budget. **ResBM** (a) needs only nearest-neighbour links, a single chain. **U-ResBM** (b) additionally needs a direct link from each stage in the first half to its mirror in the second half. The Res^{2}BM ablation (c) adds only one link to the chain, from the first stage to the last.

the uncompressed baseline, which stays high, and the compressed models, which are lower and climb with depth. The down projection, the encoder's second layer, and the decoder's first layer stay high-rank for every run, while the decoder's final layer is low-rank throughout.

B Network Communication Topology

Figure 8 shows the links each architecture requires, given seven stages. We consider the ideal case in which every edge has the same latency L and there is no jitter.

In this setting the rank schedule keeps every edge within the same latency budget. A residual that spans s stages carries sk coordinates, which is s times the load of an ordinary boundary transfer, but it is issued s stages before its destination is reached, so the larger load is matched by the longer time available to deliver it. Short and long links therefore complete within the same latency L, and no single link stalls the pipeline.

Each stage also emits two transfers at once, the ordinary boundary transfer to the next stage and the long-range residual to its mirror. Because these go to different destinations, they do not contend for the same network link.

The three architectures differ only in which links they need. ResBM uses nearest-neighbour transfers alone, so a single chain suffices. **U-ResBM** adds a direct link from each stage in the first half to its mirror in the second half. The Res^{2}BM ablation keeps only the outermost long-range skip connection, the link from the first stage to the last, adding a single edge.