大規模言語モデル事前学習の安定化

高瀬翔 [1,2]　清野舜 [1,2]　小林颯介 [3]　鈴木潤 [3]
[1]LINE ヤフー株式会社　[2]SB Intuitions 株式会社 [3] 東北大学
{sho.takase,shun.kiyono}@lycorp.co.jp
sosk@preferred.jp　jun.suzuki@tohoku.ac.jp

概要

大規模言語モデルの事前学習では損失スパイクがしばしば起きることが知られている．損失スパイクはモデルの性能を低下させ，学習が失敗してしまうこともある．この損失スパイクの原因を探るため，本研究では中間層の勾配に着目する．理論的な分析を通じて，大規模言語モデルの事前学習では LN 層が勾配爆発を引き起こすこと，および，その解決法を示す．実験を通じて LN 層の勾配爆発を抑制することで損失スパイクも抑制されることを示す．

1　はじめに

GPT をはじめ大規模言語モデル（Large Language Model，以降本稿では LLM と記す）は様々なアプリケーションの根幹となっている [1, 2, 3]．LLM は大規模な訓練データで事前学習した大量のパラメータを持つニューラル言語モデルであり，性能はパラメータ数と訓練データ量に対数比例することが知られている [4]．事前学習は大量の計算資源を必要とするため学習が失敗するリスクを極力低減することが求められている．

Transformer [5] は多くの LLM に採用されているニューラルモデルであるが，その挙動の包括的な理論的理解には至っていない．例えば，図 1 の Vanilla のように，Transformer を用いた LLM の事前学習では損失関数の値が跳ね上がり（損失スパイク），また，しばしば発散してしまう現象が知られているが [2]，なぜ発生するのかは明らかにされていない．損失スパイクへの対処として様々な手法が提案されているが [2, 6, 7]，これらの理論的な正当化もなされていないため，既存研究の報告と異なるパラメータ数のモデルなど，状況が異なる場合でも適用できるかには疑問が残る．

本研究では LLM の事前学習における損失スパイクの理論的な分析を行う．特に初期化手法が学習に与える影響について議論する．Transformer を LLM に用いる場合，Self-attention 層と FFN 層の出力ベクトルの標準偏差が小さくなる初期化を行うと学習が安定することが経験的にも理論的にも示されている [8, 6, 9, 10]．一方で，本研究では，このような初期化を行った場合には Layer Normalization（LN）層 [11] が勾配爆発を引き起こしてしまうことを示す．さらに，LN 層での勾配爆発を防ぐための要件を示し，モデルの簡易な修正でこの要件を達成できることを示す．

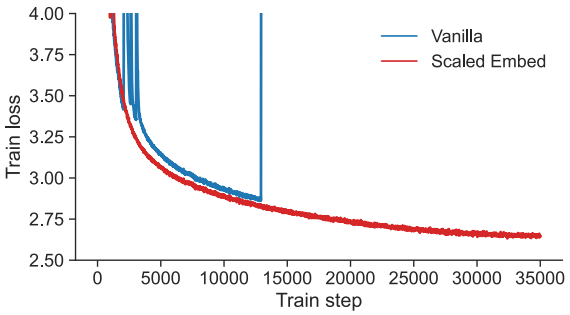


図 1　17 億パラメータの，Transformer を用いた LLM の事前学習における訓練データでの損失関数の値の例．Vanilla は学習初期の段階で損失スパイクが何度か発生し，13000 ステップで発散してしまっている．開発データでの損失関数の値は付録 A の図 4 (a) に示す．

理論的な分析を検証するため LLM の事前学習の実験を行う．上記の，LN 層での勾配爆発を防ぐ要件を満たしている場合には損失スパイクが発生せず，満たしていない場合には，仮に既存研究で損失スパイクへの対処として提案された手法であっても損失スパイクが発生してしまうことを示す．

2　議論の準備

2.1　Pre-LN Transformer

本研究では，GPT [12, 13, 1] などでも用いられている，Pre-LN Transformer [14] を対象に分析を行う．Pre-LN Transformer はオリジナルの Transformer より

も学習が安定することが示されている [14, 15, 16]. Transformer のある層への入力を $x \in \mathbb{R}^d$ としたとき，その層の出力 y は以下の式で得られる：

$$y = x' + \mathrm{FFN}(\mathrm{LN}(x')), \tag{1}$$

$$x' = x + \mathrm{Attention}(\mathrm{LN}(x)). \tag{2}$$

以降では，式 (1)，(2) の第 1 項，すなわち，x と x' を **residual branch** と呼ぶ．また，FFN と Attention の定義は以下のとおりである[1)]：

$$\mathrm{FFN}(x) = \mathcal{F}(xW_1)W_2, \tag{3}$$

$$\mathrm{Attention}(x) = \mathrm{concat}(\mathrm{head}_1(x), ..., \mathrm{head}_h(x))W_O, \tag{4}$$

$$\mathrm{head}_i(x) = \mathrm{softmax}(\frac{xW_{Qi}(XW_{Ki})^{\mathrm{T}}}{\sqrt{d_{head}}})XW_{Vi}, \tag{5}$$

ここで $\mathcal{F}$ は活性化関数[2)]，concat は入力されたベクトルを結合する関数，softmax はソフトマックス関数，$W_1 \in \mathbb{R}^{d \times d_{ffn}}$，$W_2 \in \mathbb{R}^{d_{ffn} \times d}$，$W_{Qi} \in \mathbb{R}^{d \times d_{head}}$，$W_{Ki} \in \mathbb{R}^{d \times d_{head}}$，$W_{Vi} \in \mathbb{R}^{d \times d_{head}}$，$W_O \in \mathbb{R}^{d \times d}$ は重み行列である．また，X は Self-attention 計算のために，系列分の層への入力ベクトルをまとめた行列である．

2.2 初期化手法

初期化手法はニューラルモデルの学習の挙動に多大な影響を及ぼすことが知られており，Transformer の初期化についても様々な手法が提案されている [17, 18, 19, 20]．本研究では，LLM の事前学習において広く使われている初期化手法 [8, 6, 9] を用いる前提で議論を行う．この初期化手法では $\sigma = \sqrt{2/5d}$ とし [18]，W_2 と W_O 以外の重み行列を $\mathcal{N}(0, \sigma)$ の正規分布を用いて初期化する．W_2 と W_O については標準偏差がより小さな値となるよう，$\mathcal{N}(0, \sigma/\sqrt{2N})$ を用いて初期化を行う．なお，N は層の数である．

3 事前学習の不安定性の分析

我々は勾配爆発が学習を不安定にし，損失スパイクを引き起こしていると考えている．本節では，LN 層が勾配爆発を引き起こすこと，また，勾配爆発を防ぐ手法を紹介する．

3.1 LN 層での勾配爆発

Xiong らは下記式のように，LN の勾配のノルムは入力ベクトルのノルムに依存することを証明した [14][3)]：

$$||\frac{\delta \mathrm{LN}(x)}{\delta x}||_2 = \mathcal{O}(\frac{\sqrt{d}}{||x||_2}), \tag{6}$$

ここで，x が正規分布に従うとすると，x の平均は 0 であり x のノルムは標準偏差に比例するため，標準偏差を考えればノルムが得られる．具体的には，$\mathrm{std}(x)$ を x の標準偏差とすると，$||x||_2 = \sqrt{d}\,\mathrm{std}(x)$ となる．よって上記の式は下記のように変形できる：

$$||\frac{\delta \mathrm{LN}(x)}{\delta x}||_2 = \mathcal{O}(\frac{1}{\mathrm{std}(x)}). \tag{7}$$

すなわち，$\mathrm{std}(x) \ll 1$ の場合に LN 層は勾配のノルムを著しく増加させる．勾配がどこで増加しているかを推定するため，LN に入力されるベクトルの標準偏差を調べていく．

2.1 節で記したように，各中間層の出力が LN 層への入力となる．すなわち，式 (1)，(2) のように，residual branch と $\mathrm{FFN}(x)$ もしくは $\mathrm{Attention}(x)$ の和が次の層の LN 層への入力となる．ここで，$\mathrm{var}(x)$ をベクトル x の分散とすると，正規分布に従うベクトル同士の和については下記が成り立つ：

$$\mathrm{var}(x + y) = \mathrm{var}(x) + \mathrm{var}(y). \tag{8}$$

従って，$\mathrm{FFN}(x)$ および $\mathrm{Attention}(x)$ で得られるベクトルもまた正規分布に従うと仮定した場合[4)]，各層の出力の分散については，residual branch，$\mathrm{FFN}(x)$，$\mathrm{Attention}(x)$ の分散を考えれば良い．第 1 層目の residual branch は埋め込み表現であり，埋め込み表現は $\mathcal{N}(0, \sigma)$ で初期化されている．2.2 節より，$\sigma = \sqrt{2/5d} \ll 1$ であるため，$\mathrm{std}(\mathrm{FFN}(x))$ と $\mathrm{std}(\mathrm{Attention}(x))$ が 1 よりも十分小さければ，式 (8) より各層の出力の標準偏差も 1 より十分小さく，LN 層によって勾配が爆発する．

学習初期の $\mathrm{var}(\mathrm{FFN}(x))$ は以下の式で得られる：

$$\mathrm{var}(\mathrm{FFN}(x)) = d_{ffn}\, d\, \mathrm{var}(x)\, \mathrm{var}(W_1)\, \mathrm{var}(W_2). \tag{9}$$

なお，式 (1) にあるように FFN の計算前には LN が適用されるので $\mathrm{var}(x) = 1$ である．多くの研究で採用されている値と同様に $d_{ffn} = 4d$ とし，$\mathrm{var}(W_1)$ と $\mathrm{var}(W_2)$ については 2.2 節での値を代入すると，

$$\mathrm{var}(\mathrm{FFN}(x)) = 4d\, d\, \frac{2}{5d}\frac{2}{5d}\frac{1}{2N},$$

$$= \frac{8}{25N}. \tag{10}$$

1) 簡略化するために本稿ではバイアス項を抜いている．
2) 議論を単純にするために，本稿では恒等写像を仮定する．
3) Xiong らの証明は RMSNorm [21] にも適用できるため，本稿での議論は LN を RMSNorm に代えても成立する．
4) 各パラメータを正規分布で初期化しているため，$\mathcal{F}$ を恒等写像と置き，$\mathrm{Attention}(x)$ のヘッドが 1 の場合にはこの仮定は成立する．実際のモデルにおいてどの程度現実的な仮定となっているかの検証は今後の課題である．

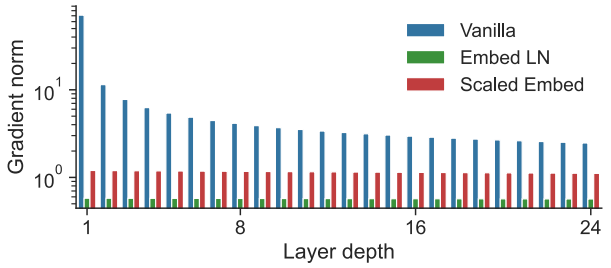


図 2　事前学習初期の Transformer の各層の勾配のノルム．

一般に LLM においては $N > 10$ であるため，$\mathrm{std}(\mathrm{FFN}(x)) = \sqrt{8/25N} \ll 1$ となる．

次に $\mathrm{var}(\mathrm{Attention}(x))$ について考える．まず $\mathrm{var}(\mathrm{head}_i(x))$ について考える．$Z = \frac{xW_{Qi}(XW_{Ki})^{\mathrm{T}}}{\sqrt{d_{head}}}$ とすると $\mathrm{var}(\mathrm{softmax}(\frac{xW_{Qi}(XW_{Ki})^{\mathrm{T}}}{\sqrt{d_{head}}})X) = \mathrm{var}(\mathrm{softmax}(Z)X)$ と書ける．X は LN を適用されたベクトルをまとめたものであるため，$\mathrm{var}(\mathrm{softmax}(Z)X)$ は $\mathrm{softmax}(Z)$ が 1-hot ベクトルである場合に最大値である 1 となる[5]．従って，式 (5) の分散は以下で得られる：

$$\begin{aligned}\mathrm{var}(\mathrm{head}_i(x)) &= \mathrm{var}(\mathrm{softmax}(Z)X)\ d\ \mathrm{var}(W_{Vi}),\\ &= \mathrm{var}(\mathrm{softmax}(Z)X)\ d\ \frac{2}{5d},\\ &< \frac{2}{5}.\end{aligned} \tag{11}$$

従って，$\mathrm{var}(\mathrm{Attention}(x))$ は以下の式で得られる：

$$\begin{aligned}\mathrm{var}(\mathrm{Attention}(x)) &= \mathrm{var}(\mathrm{head}(x))\ d\ \mathrm{var}(W_O),\\ &= \mathrm{var}(\mathrm{head}(x))\ d\ \frac{2}{5d}\frac{1}{2N},\\ &= \mathrm{var}(\mathrm{head}(x))\ \frac{1}{5N},\\ &< \frac{2}{25N}.\end{aligned} \tag{12}$$

$\mathrm{std}(\mathrm{FFN}(x))$ の際と同様に $N > 10$ であるため，$\mathrm{std}(\mathrm{Attention}(x)) < \sqrt{2/25N} \ll 1$ となる．これらから，特に浅い層において[6]，式 (1)，(2) の出力ベクトルの標準偏差は 1 より十分小さく，LN 層によって勾配が爆発する．

3.2　LN 層の勾配の安定化手法

LN 層での勾配爆発を防ぐためには LN 層へ入力するベクトルの標準偏差が 1 に近い値である必要がある．これを達成するためには次のいずれかの方法がある：1. residual branch の標準偏差を大きくする，2. $\mathrm{FFN}(x)$ と Attention(x) の標準偏差を大きくする．しかしながら，2 の場合には層が深くなるに連れての各層の出力のノルムの増加が著しく，これによって LN 層に起因しない勾配爆発が生じてしまう．悪影響を及ぼすことなく LN 層の勾配爆発を防ぐために，1 を選択する．

第 1 層目の residual branch は埋め込み表現であるため，埋め込み表現の標準偏差を 1 に近づければ residual branch の標準偏差も 1 に近づく．これを達成する手法はいくつか考えられるが[7]，本稿では 2 種，Scaled Embed と Embed LN を紹介する．Scaled Embed は適切な値で埋め込み表現をスケーリングするものである．例えば，埋め込み表現に $\sqrt{d}$ を掛け合わせると[8]，埋め込み表現の標準偏差は $\sqrt{2/5}$ となる．Embed LN は LN を埋め込み表現に適用するもので，既存研究でも損失スパイクを防いだと報告されている [6]．学習初期での各層の勾配のノルムを図 2 に示す．この図から，何も対処をしていない場合（Vanilla）は層が浅くなるに連れて勾配が大きく増大しているが，Scaled Embed と Embed LN は勾配のノルムが一定であることが分かる．すなわち，Scaled Embed と Embed LN は LN 層による勾配爆発を防いでいることが分かる．

4　実験

本稿での理論分析について，実際に LLM の事前学習を通して検証する．すなわち，勾配爆発を抑える要件を満たす手法は損失スパイクが発生しないこと，要件を満たしていない手法は既存研究では損失スパイクの抑制に効果があるとされていてもスパイクを引き起こしてしまうことを示す．

4.1　データセット

事前学習の訓練・開発データとしては Common Crawl[9]から抽出した英語コーパスである，C4 [25] を用いる．語彙は Byte Pair Encoding（BPE）[26] で構築されている，GPT-2 [13] の語彙を用いる．事前学習したモデルの評価には WikiText [27] と

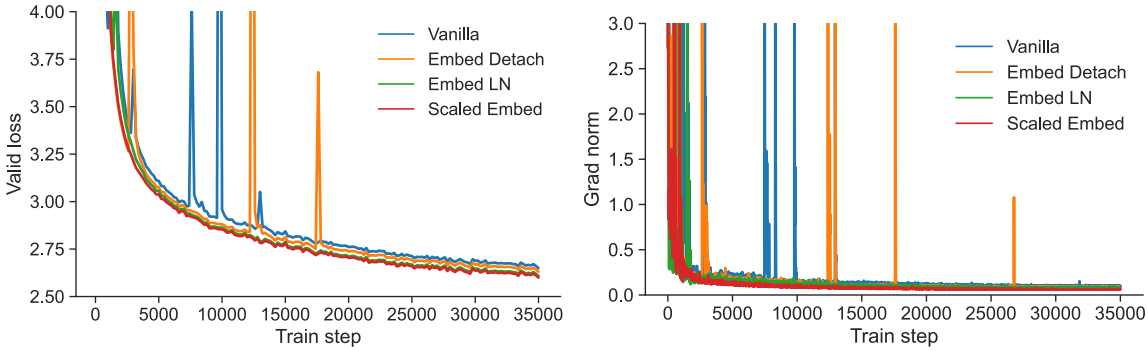
5）　なお，$\mathrm{softmax}(Z)$ が一様分布となる場合に最小値となり，入力系列が長いほど値が小さくなる．Li らは学習初期の入力系列は短い方が安定すると報告しているが [22]，本稿の議論から，$\mathrm{var}(\mathrm{softmax}(Z)X)$ の値が小さくなりすぎないように抑制する効果があるためと推測される．

6）　$\mathrm{FFN}(x)$ と $\mathrm{Attention}(x)$ は層が深くなるごとに足されていくため，出力の標準偏差は層が深くなるほど大きくなっていく．例えば，最終層では N 回足されているため，式 (10) と (12) の分母の N を除去できる．

7）　例えば埋め込み表現と出力層のパラメータ共有 [23, 24] を行わない場合，埋め込み表現の初期化における $\mathcal{N}(0, \sigma)$ について $\sigma = 1$ とすれば良い．本稿では初期化は 2.2 節で紹介したものを用いる前提で議論を進める．

8）　この手続きはオリジナルの Transformer では行われていたが [5]，近年の実装では削除されている．

9）　https://commoncrawl.org/



(a) 開発データでの損失関数の値．　(b) 勾配のノルム．

図 3　各手法の更新回数に対する開発データでの損失関数の値と勾配のノルム．

LAMBADA [28] データセットを用いてパープレキシティを計算する．

4.2　モデル設定

2 節で記したように，Pre-LN Transformer を用い，LLM の事前学習で広く使われている手法 [8, 6] で初期化する．モデルパラメータは 17 億とし，層数 $N = 24$，各層の次元数 $d = 2304$ とする．入力の系列長は 2048 とし，勾配ノルムを 1.0 でクリッピングする．学習率 (lr) は 5.0×10^{-4} とする[10]．詳細なハイパーパラメータは付録 B に記す．

4.3　比較手法

3 節で言及した **Vanilla**，**Embed LN**，**Scaled Embed** に加え，**Embed Detach** を比較手法とする．Embed Detach では下記式のように，埋め込み表現の勾配を部分的に計算グラフから切り離すことで勾配を縮小する [29]：

$$\text{Embed} = \gamma\text{Embed} + (1-\gamma)\text{Detach}(\text{Embed}), \qquad (13)$$

ここで γ はハイパーパラメータであり，Detach は入力を計算グラフから切り離す関数である．Zeng らは Embed Detach を LLM の事前学習に適用することで学習が安定したと報告しているが，この手法は本稿で説明した勾配爆発を防げてはいないため，本質的には損失スパイクを抑制できていないはずである．本実験では Zeng らと同様に $\gamma = 0.1$ とする．

4.4　結果

図 3 に更新回数に対する各手法の開発データでの損失関数の値と勾配のノルムを示す．この図から分かるように，Vanilla と Embed Detach では損失スパイクおよび勾配のノルムのスパイクが起きているが，Embed LN と Scaled Embed では起こっていない．すなわち，3 節で示した勾配爆発を防ぐ要件を満たす手法では損失スパイクが起きておらず，勾配爆発の抑制が LLM の事前学習の安定させることが分かる．

表 1　各手法のパープレキシティ．

Model	WikiText ↓	LAMBADA ↓
Vanilla	22.58	15.22
Embed Detach	22.00	13.88
Embed LN	**21.29**	13.00
Scaled Embed	**21.29**	**12.53**

表 1 に各手法の WikiText と LAMBADA でのパープレキシティを示す．この表から，Embed LN と Scaled Embed は Vanilla や Embed Detach よりも高い性能を達成していることが分かる．この結果は，損失スパイクを防ぐことは学習したモデルの性能向上にも貢献すると示唆している．また，Embed LN と Scaled Embed は同等の性能を達成していることから，損失スパイクを抑制していれば，手法間には性能に有意な差はないと考えられる．

5　おわりに

本研究では LLM の事前学習においてしばしば発生する，損失スパイクの原因を分析した．勾配に着目し，LLM の事前学習では LN 層が勾配を増大させていること，residual branch の標準偏差を 1 に近づけることで勾配の増加を抑制できることを理論的に示した．実験を通して，residual branch の標準偏差を 1 に近づける手法，すなわち，Embed LN と Scaled Embed は実際に損失スパイクや勾配のノルムのスパイクを抑制し，LLM の事前学習を安定化させることを示した．

10)　学習率やパラメータ数を変えた実験を付録 A に記す．

謝辞

本稿の執筆にあたり理研 AIP の Benjamin Heinzerling 氏に有用な助言をいただきました．感謝いたします．

参考文献

[1] T. Brown, B. Mann, N. Ryder, et al. Language models are few-shot learners. In **Advances in Neural Information Processing Systems 33 (NeurIPS)**, pages 1877–1901, 2020.

[2] A. Chowdhery, S. Narang, J. Devlin, et al. PaLM: Scaling language modeling with pathways, 2022.

[3] H. Touvron, L. Martin, K. Stone, et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

[4] J. Kaplan, S. McCandlish, T. Henighan, et al. Scaling laws for neural language models, 2020.

[5] A. Vaswani, N. Shazeer, N. Parmar, et al. Attention is all you need. In **Advances in Neural Information Processing Systems 30 (NIPS)**, pages 5998–6008. 2017.

[6] T. Le Scao, T. Wang, D. Hesslow, et al. What language model to train if you have one million GPU hours? In **Findings of the Association for Computational Linguistics: EMNLP 2022**, pages 765–782, 2022.

[7] A. Zeng, X. Liu, Z. Du, et al. GLM-130b: An open bilingual pre-trained model. In **The Eleventh International Conference on Learning Representations (ICLR)**, 2023.

[8] M. Shoeybi, M. Patwary, R. Puri, et al. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.

[9] S. Biderman, H. Schoelkopf, Q. Anthony, et al. Pythia: A suite for analyzing large language models across training and scaling. In **Proceedings of the 40th International Conference on Machine Learning (ICML)**, pages 2397—2430, 2023.

[10] S. Takase, S. Kiyono, S. Kobayashi, and J. Suzuki. Spike no more: Stabilizing the pre-training of large language models, 2023.

[11] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016.

[12] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

[13] A. Radford, J. Wu, R. Child, et al. Language models are unsupervised multitask learners. 2019.

[14] R. Xiong, Y. Yang, D. He, et al. On layer normalization in the transformer architecture. In **Proceedings of the 37th International Conference on Machine Learning (ICML)**, pages 10524–10533, 2020.

[15] L. Liu, X. Liu, J. Gao, W. Chen, and J. Han. Understanding the difficulty of training transformers. In **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)**, pages 5747–5763, 2020.

[16] S. Takase, S. Kiyono, S. Kobayashi, and J. Suzuki. B2T connection: Serving stability and performance in deep transformers. In **Findings of the Association for Computational Linguistics: ACL 2023**, pages 3078–3095, 2023.

[17] H. Zhang, Y. N. Dauphin, and T. Ma. Fixup initialization: Residual learning without normalization. In **Proceedings of the 7th International Conference on Learning Representations (ICLR)**, 2019.

[18] T. Q. Nguyen and J. Salazar. Transformers without tears: Improving the normalization of self-attention. In **Proceedings of the 16th International Conference on Spoken Language Translation (IWSLT)**, 2019.

[19] B. Zhang, I. Titov, and R. Sennrich. Improving deep transformer with depth-scaled initialization and merged attention. In **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)**, pages 898–909, 2019.

[20] H. Wang, S. Ma, L. Dong, et al. Deepnet: Scaling transformers to 1,000 layers, 2022.

[21] B. Zhang and R. Sennrich. Root mean square layer normalization. In **Advances in Neural Information Processing Systems 32 (NeurIPS)**, 2019.

[22] C. Li, M. Zhang, and Y. He. The stability-efficiency dilemma: Investigating sequence length warmup for training gpt models. In **Advances in Neural Information Processing Systems 35 (NeurIPS)**, pages 26736–26750, 2022.

[23] O. Press and L. Wolf. Using the output embedding to improve language models. In **Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)**, pages 157–163, 2017.

[24] H. Inan, K. Khosravi, and R. Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In **Proceedings of the 5th International Conference on Learning Representations (ICLR)**, 2017.

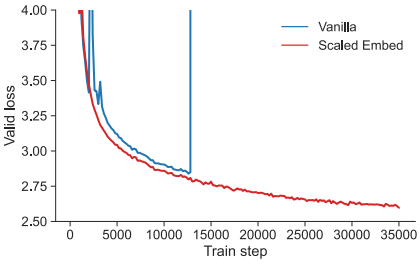
[25] C. Raffel, N. Shazeer, A. Roberts, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. **Journal of Machine Learning Research**, 21(140):1–67, 2020.

[26] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)**, pages 1715–1725, 2016.

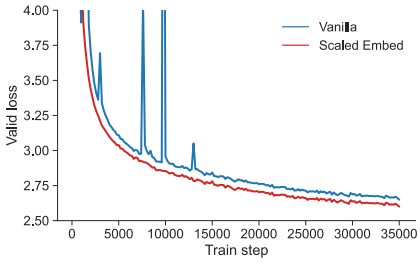
[27] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer Sentinel Mixture Models. In **Proceedings of the 5th International Conference on Learning Representations (ICLR)**, 2017.

[28] D. Paperno, G. Kruszewski, A. Lazaridou, et al. The LAMBADA dataset: Word prediction requiring a broad discourse context. In **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)**, pages 1525–1534, 2016.

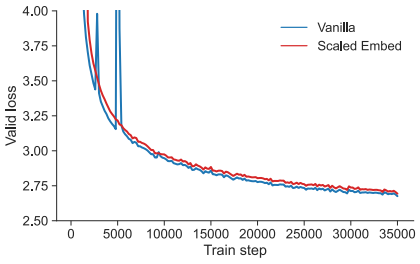
[29] M. Ding, Z. Yang, W. Hong, et al. Cogview: Mastering text-to-image generation via transformers. In **Advances in Neural Information Processing Systems 34 (NeurIPS)**, pages 19822–19835, 2021.



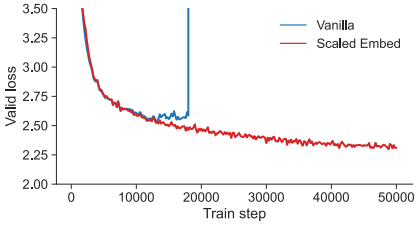
(a) 17 億パラメータ，lr = 1.0×10^{-3}.



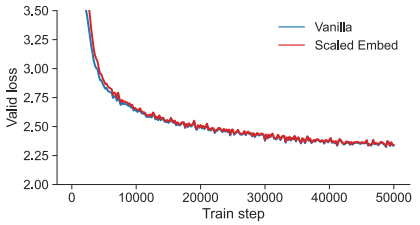
(b) 17 億パラメータ，lr = 5.0×10^{-4}.



(c) 17 億パラメータ，lr = 1.0×10^{-4}.



(d) 130 億パラメータ，lr = 3.0×10^{-4}.



(e) 130 億パラメータ，lr = 1.0×10^{-4}.

図 4　17 億パラメータおよび 130 億パラメータで学習率を変えた際の各手法の開発データでの損失関数の値.

表 2　17 億パラメータの各手法の各学習率におけるパープレキシティ.

	WikiText ↓			LAMBADA ↓		
Model	lr = 1.0×10^{-3}	lr = 5.0×10^{-4}	lr = 1.0×10^{-4}	lr = 1.0×10^{-3}	lr = 5.0×10^{-4}	lr = 1.0×10^{-4}
Vanilla	N/A	22.58	**23.54**	N/A	15.22	16.17
Scaled Embed	**20.95**	**21.29**	23.78	**12.26**	**12.53**	**15.39**

表 3　130 億パラメータの各手法の各学習率におけるパープレキシティ.

	WikiText ↓		LAMBADA ↓	
Model	lr = 3.0×10^{-4}	lr = 1.0×10^{-4}	lr = 3.0×10^{-4}	lr = 1.0×10^{-4}
Vanilla	N/A	**15.12**	N/A	**6.50**
Scaled Embed	**14.47**	15.25	**5.97**	6.53

A　パラメータ数・学習率を変えた実験

本節では 4 節での 17 億パラメータ，学習率 5.0×10^{-4} に加え，17 億パラメータで学習率を 1.0×10^{-3}，1.0×10^{-4} とした場合，および，130 億パラメータでの実験を行う．130 億パラメータは実際の LLM の事前学習の状況により近くなるよう，バッチサイズを大きな値とし，Adam の $\beta_2 = 0.95$ とした．$\beta_2 = 0.95$ は事前学習を安定にすると報告されている値である．また，学習率は Llama 2 [3] で用いられている 3.0×10^{-4} と，より低い値である 1.0×10^{-4} を用いた．4 節と同様に，訓練・開発データは C4 を，語彙には GPT-2 のものを用いた．計算資源の都合上，ここでは Vanilla と Scaled Embed のみを対象に実験を行う．

図 4 に各手法の各設定での更新回数に対する開発データ上での損失関数の値を示す．17 億パラメータについてである，図 (a), (b), (c) から明らかなように，Vanilla では学習率を大きくするほど損失スパイクが頻繁に発生するようになっている．特に，17 億パラメータのモデルを学習率 1.0×10^{-3} で学習した際には，Vanilla では損失関数の値が発散してしまい，学習が失敗してしまっている．一方で，Scaled Embed については学習が安定しており，損失関数の値は更新回数に応じて一貫して下がっている．130 億パラメータについては，$\beta_2 = 0.95$ としているからか，Vanilla でも学習率 1.0×10^{-4} で損失スパイクは見られない．しかしながら，高い学習率，すなわち，lr = 3.0×10^{-4} では損失関数の値が発散してしまっており，学習が失敗してしまっている．この結果から，$\beta_2 = 0.95$ のような学習を安定化させると報告されている状況においても，より安定した学習のために，residual branch の標準偏差を 1 に近づけることは必要と言える．表 2，表 3 に 17 億パラメータと 130 億パラメータの WikiText と LAMBADA データセットでのパープレキシティを示す．これらの表から，17 億パラメータ，130 億パラメータ共に，学習率を変えた場合でも Scaled Embed は Vanilla と同等か，より高い性能を達成できていることが分かる．

表 4　実験に用いたハイパーパラメータ.

パラメータ数	17 億	130 億
層数	24	40
次元数	2304	5120
ヘッドの数	24	40
バッチサイズ	528	2097152
更新回数	35000	50000
Adam β_1	0.9	0.9
Adam β_2	0.999	0.95
Weight decay	0.01	0.01

B　ハイパーパラメータ

表 4 に実験に用いたハイパーパラメータを示す.