A SUPPLEMENTARY MATERIAL

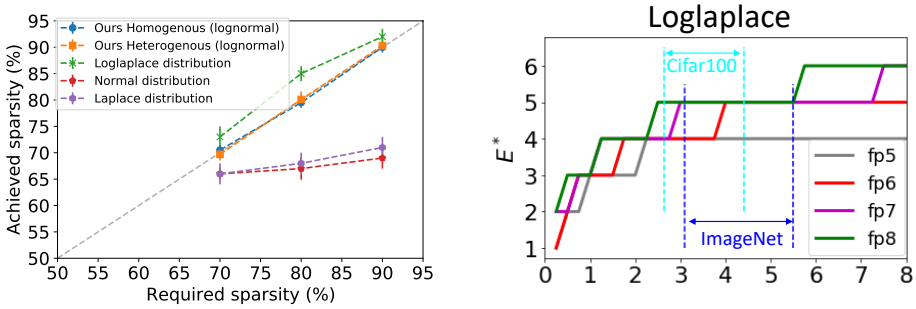
A.1 NEURAL GRADIENTS DISTRIBUTION — DISCUSSION

In order to analyze how well neural gradients fit to a lognormal distribution, we measure in Table A.1 the Kolmogorov-Smirnov test (Smirnov, 1948) with different distributions in different models and datasets, similar to Table 1. Notice that lognormal and loglaplace distributions achieve much higher fit than the other distribution, with a small statistical advantage of the lognormal distribution.

In order to emphasize of the choice of the lognormal distribution as the best fit, we show in Fig. A.1a the same experiment described in Section 6 with additional distributions, i.e solve equation Eq. (6) with different distribution assumptions and check the achieved sparsity in that case. We can empirically notice that the lognormal distribution is a better fit for the neural gradients than the other distributions. Additionaly, we show in Fig. A.1b the FP format obtained by solving Eq. (3) with loglaplace distribution (Similar to Fig. 3b and Fig. 3c). Notice this leads to sub-optimal format - for example in Cifar100 FP6 format, loglaplace assumption leads to 1-5-0 as the optimal format, which is sub-optimal as shown in Table 3.

Table A.1: Mean ($\pm$ std) over all layers over time of Kolmogorov-Smirnov test on different models and datasets. Notice the lognormal distribution get the higher fit across all models

Model-Dataset	Laplace	Normal	Uniform	Cauchy	Loglaplace	Lognormal
BERT-CoLA	0.46±0.02	0.46±0.02	0.96±0.02	0.46±0.01	0.07±0.012	**0.05±0.002**
BERT- MRPC	0.41±0.03	0.39±0.04	0.87±0.04	0.41±0.03	0.07±0.015	**0.04±0.002**
ResNet18-ImageNet	0.32±0.1	0.38±0.1	0.77±0.1	0.37±0.15	0.05±0.001	**0.02±0.002**
MobileNetV2-ImageNet	0.22±0.09	0.22±0.09	0.69±0.12	0.26±0.06	0.09±0.02	**0.07±0.003**
VGG16-ImageNet	0.37±0.09	0.35±0.08	0.8±0.1	0.39±0.08	0.08±0.032	**0.06±0.002**
DenseNet121-ImageNet	0.32±0.1	0.33±0.1	0.76±0.06	0.38±0.09	0.075±0.01	**0.05±0.001**



(a) (b)

Figure A.1: **(a)**Comparison of the required and achieved sparsity of stochastic pruning with different distribution assumptions. Notice that lognormal distribution achieves the minimum deviation from the requires sparsity. This emphasizes the goodness of fit of the neural gradients to a lognormal distribution. **(b)** Ideal bit allocation assuming a loglaplace distribution leads to sub-optimal formats. For example, FP6 turns out to have a bit-allocation of 1-5-0, which is inferior to 1-4-1 (the optimal format assuming log-normal distribution), as shown in Table 3.

A.2 NEURAL GRADIENTS BEFORE AND AFTER BATCH-NORM

A neural network is a composition of different blocks that include linear and non-linear functions. In CNNs the most common block contains the trio Conv-BN-ReLU. Consider a CNN with L layers, the output, $\mathcal{A}_{i;3}$, of layer $i \in \{1, ..., L\}$ is:

$$\textbf{[Conv]}\ \mathcal{A}_{i;1} = W_i \times \mathcal{A}_{i-1;3} \quad \textbf{[BN]}\ \mathcal{A}_{i;2} = \mathrm{BN}_{\beta,\gamma}(\mathcal{A}_{i;1}) \quad \textbf{[ReLU]}\ \mathcal{A}_{i;3} = \max(0, \mathcal{A}_{i;2})\,, \quad \text{(A.1)}$$

And the corresponding gradients:

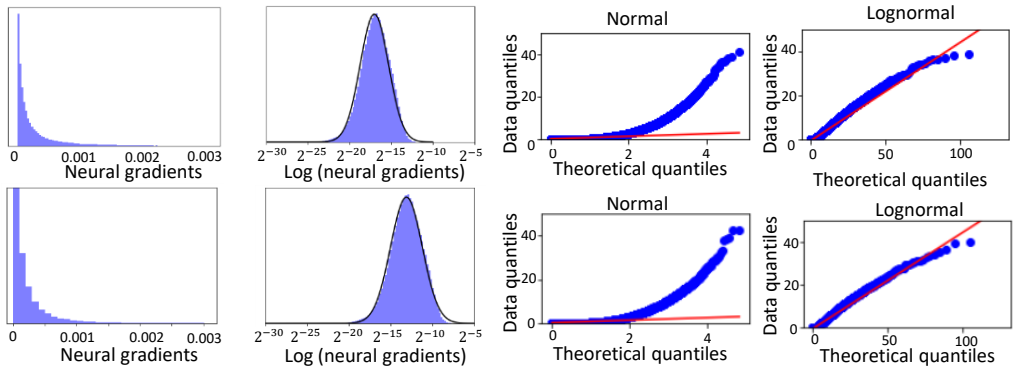


Figure A.2: Identifying the distribution of neural gradients (normal vs. lognormal) for additional layers in ResNet18 - ImageNet dataset, similar to Fig. 2.

Table A.2: Mean ± std (p-value) over all layers at different epochs of KS test on different models and datasets for normal and lognormal distribution (Similar to Table 1). Notice the lognormal distribution gets the higher fit across all models.

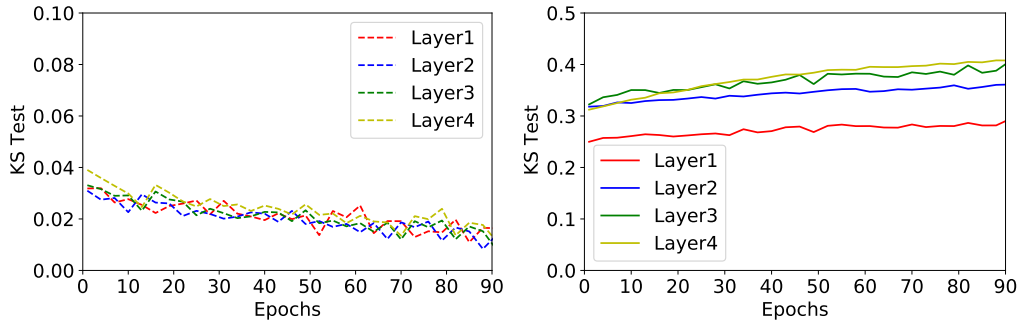
Epoch	Distribution	Model (Dataset)					
		BERT (CoLa)	BERT (MRPC)	ResNet18 (ImageNet)	MobileNetV2 (ImageNet)	VGG16 (ImageNet)	DenseNet121 (ImageNet)
10	Normal	0.38±0.03 ($3 \cdot 10^{-4}$)	0.35±0.03 ($5 \cdot 10^{-5}$)	0.29±0.12 ($3 \cdot 10^{-6}$)	0.18±0.09 ($8 \cdot 10^{-5}$)	0.33±0.07 ($3 \cdot 10^{-6}$)	0.32±0.09 ($8 \cdot 10^{-4}$)
	Lognormal	**0.07±0.003** (0.25)	**0.04±0.001** (0.21)	**0.04±0.001** (0.23)	**0.09±0.003** (0.14)	**0.04±0.001** (0.28)	**0.04±0.001** (0.26)
30	Normal	0.46±0.03 ($3 \cdot 10^{-5}$)	0.35±0.03 ($4 \cdot 10^{-6}$)	0.37±0.12 ($2 \cdot 10^{-4}$)	0.24±0.07 ($3 \cdot 10^{-6}$)	0.33±0.09 ($5 \cdot 10^{-6}$)	0.33±0.12 ($5 \cdot 10^{-5}$)
	Lognormal	**0.08±0.002** (0.28)	**0.03±0.002** (0.23)	**0.01±0.005** (0.26)	**0.09±0.002** (0.18)	**0.04±0.001** (0.31)	**0.04±0.003** (0.29)
50	Normal	0.45±0.01 ($5 \cdot 10^{-4}$)	0.34±0.02 ($6 \cdot 10^{-5}$)	0.37±0.08 ($4 \cdot 10^{-5}$)	0.25±0.1 ($3 \cdot 10^{-6}$)	0.34±0.09 ($3 \cdot 10^{-6}$)	0.35±0.07 ($5 \cdot 10^{-5}$)
	Lognormal	**0.08±0.002** (0.26)	**0.05±0.001** (0.21)	**0.02±0.001** (0.26)	**0.08±0.003** (0.17)	**0.07±0.003** (0.35)	**0.05±0.001** (0.33)
70	Normal	0.47±0.01 ($8 \cdot 10^{-5}$)	0.38±0.05 ($6 \cdot 10^{-4}$)	0.41±0.12 ($5 \cdot 10^{-5}$)	0.21±0.07 ($4 \cdot 10^{-5}$)	0.36±0.07 ($3 \cdot 10^{-6}$)	0.31±0.08 ($5 \cdot 10^{-5}$)
	Lognormal	**0.07±0.001** (0.27)	**0.03±0.001** (0.25)	**0.02±0.005** (0.32)	**0.06±0.004** (0.22)	**0.03±0.001** (0.29)	**0.08±0.002** (0.28)
90	Normal	0.46±0.01 ($2 \cdot 10^{-4}$)	0.38±0.03 ($5 \cdot 10^{-5}$)	0.34±0.2 ($3 \cdot 10^{-6}$)	0.25±0.08 ($5 \cdot 10^{-6}$)	0.37±0.05 ($3 \cdot 10^{-6}$)	0.35±0.08 ($5 \cdot 10^{-5}$)
	Lognormal	**0.04±0.001** (0.25)	**0.02±0.002** (0.23)	**0.02±0.003** (0.26)	**0.08±0.005** (0.17)	**0.07±0.002** (0.31)	**0.05±0.001** (0.26)

$$\textbf{[ReLU]} \quad \nabla \mathcal{A}_{i;2} = \nabla \mathcal{A}_{i;3} \cdot \delta(A_{i;2} > 0) \tag{A.2}$$

$$\begin{aligned} \textbf{[BN]} \quad \nabla \mathcal{A}_{i;1} &= \frac{\gamma_i}{\sqrt{\sigma^2 + \epsilon}} \left[\nabla \mathcal{A}_{i;2} - \frac{1}{N} \left(\nabla \gamma \left(\frac{\mathcal{A}_{i;1} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \right) + \nabla \beta \right) \right] \\ \nabla \beta &= \sum (\nabla \mathcal{A}_{i;2}) \quad \nabla \gamma = \sum \left(\nabla \mathcal{A}_{i;2} \times \frac{\mathcal{A}_{i;1} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \right) \end{aligned} \tag{A.3}$$

$$\textbf{[Conv]} \quad \nabla \mathcal{A}_{i-1;3} = \nabla \mathcal{A}_{i;1} \times W_i^T \quad \nabla W_i = \nabla \mathcal{A}_{i;1} \times \mathcal{A}_{i-1;3} \tag{A.4}$$

where $\times$ defines a convolution operation, W_i is the weights matrix, and BN is a batch-norm operation. The corresponding gradients are $\nabla \mathcal{A}_{i;1-3}, \nabla W_i, \nabla \beta$ and $\nabla \gamma$



(a) (b)

Figure A.3: KS Test in different layers across the training of ResNet-18 ImageNet dataset for lognormal (a) normal (b) distribution. Notice that the high fit of lognormal distribution to the neural gradients is noticeable across all training process.

The distribution of $\nabla\mathcal{A}_{i-1;3}$ fit to lognormal distribution as shown in the main paper. The distribution of $\nabla\mathcal{A}_{i;1}$ can be approximated as a bi-modal lognormal distribution. The two modes stem from two different components of $\nabla\mathcal{A}_{i;2}$: the left mode originates from the zero-valued elements (that are abundant because of the ReLU) and the right mode from the rest of $\nabla\mathcal{A}_{i;2}$. We split the elements of $\nabla\mathcal{A}_{i;1}$ according to the value of the same element in $\nabla\mathcal{A}_{i;2}$ (zero or non-zero), then apply in each of them Kolmogorov-Smirnov test.

The two modes can be separated and analyzed by splitting the entries of $\nabla\mathcal{A}_{i;1}$ according to the value of the same entries in $\nabla\mathcal{A}_{i;2}$ (zero or non-zero); see the results of the Kolmogorov-Smirnov test to these distributions in Table A.3 - notice each of them can best be described as a lognormal distribution. Examples can be seen in Fig. A.4.

To estimate $\nabla\mathcal{A}_{i;1}$, we divide the data of $\nabla\mathcal{A}_{i;2}$ to zero and non-zero elements. Then, we apply Kolmogorov-Smirnov test (Smirnov, 1948) to each of them, and show in Table A.3 they fit to lognormal distribution - obtaining a total of bi modal lognormal distribution in $\nabla\mathcal{A}_{i;1}$.

Table A.3: Mean ($\pm$ std) over all layers of Kolmogorov-Smirnov test on $\nabla\mathcal{A}_{i;1}$ over the zero and non zero elements of $\nabla\mathcal{A}_{i;2}$ ResNet18 on ImageNet

Distribution	Laplace	Normal	Uniform	Cauchy	Logistic	Loglaplace	Lognormal
zeros	0.43±0.13	0.41±0.09	0.93±0.19	0.44±0.12	0.53±0.13	0.1±0.03	**0.08±0.03**
non-zeros	0.48±0.11	0.49±0.13	0.86±0.09	0.43±0.11	0.51±0.14	0.04±0.012	**0.02±0.01**

Histograms. In Fig. A.4 we show histograms of $\nabla\mathcal{A}_{i;1}$ divided to the zero and non-zero elements from $\nabla\mathcal{A}_{i;2}$ in ResNet18, ImageNet which as shown in Appendix A.2 each of them fit to lognormal distribution. Additionally we show histograms of different layers of Transformer (Vaswani et al., 2017), DenseNet121 and Vgg16 which at log-scale following to normal distribution.

A.3 From sparsity to the threshold — the full solution

$$
\begin{aligned}
S &= \mathbb{E}_{\varepsilon}\left[\frac{1}{2}+\frac{1}{2}\text{erf}\left(\frac{\ln\left(\alpha\cdot\varepsilon\right)-\mu}{\sqrt{2}\sigma}\right)\right] \underset{\varepsilon'=\frac{\varepsilon}{e^{\mu}}}{=} \int_{0}^{\frac{\alpha}{e^{\mu}}}\left[\frac{1}{2}+\frac{1}{2}\text{erf}\left(\frac{\ln\left(\tau\right)}{\sqrt{2}\sigma}\right)\right]\frac{e^{\mu}}{\alpha}\,d\tau \\
&= \frac{1}{2}+\frac{e^{\mu}}{2\alpha}\left[e^{\frac{\sigma^2}{2}}\text{erf}\left(\frac{\sigma^2-\ln(\tau)}{\sqrt{2}\sigma}\right)+\tau\text{erf}\left(\frac{\ln(\tau)}{\sqrt{2}\sigma}\right)\right]\Big|_{0}^{\frac{\alpha}{e^{\mu}}} \\
&= \frac{1}{2}+\frac{e^{\mu}}{2\alpha}\left[e^{\frac{\sigma^2}{2}}\text{erf}\left(\frac{\sigma}{\sqrt{2}}-\frac{\ln\left(\frac{\alpha}{e^{\mu}}\right)}{\sqrt{2}\sigma}\right)+\frac{\alpha}{e^{\mu}}\cdot\text{erf}\left(\frac{\ln\left(\frac{\alpha}{e^{\mu}}\right)}{\sqrt{2}\sigma}\right)-e^{\frac{\sigma^2}{2}}\right]
\end{aligned}
\tag{A.5}
$$

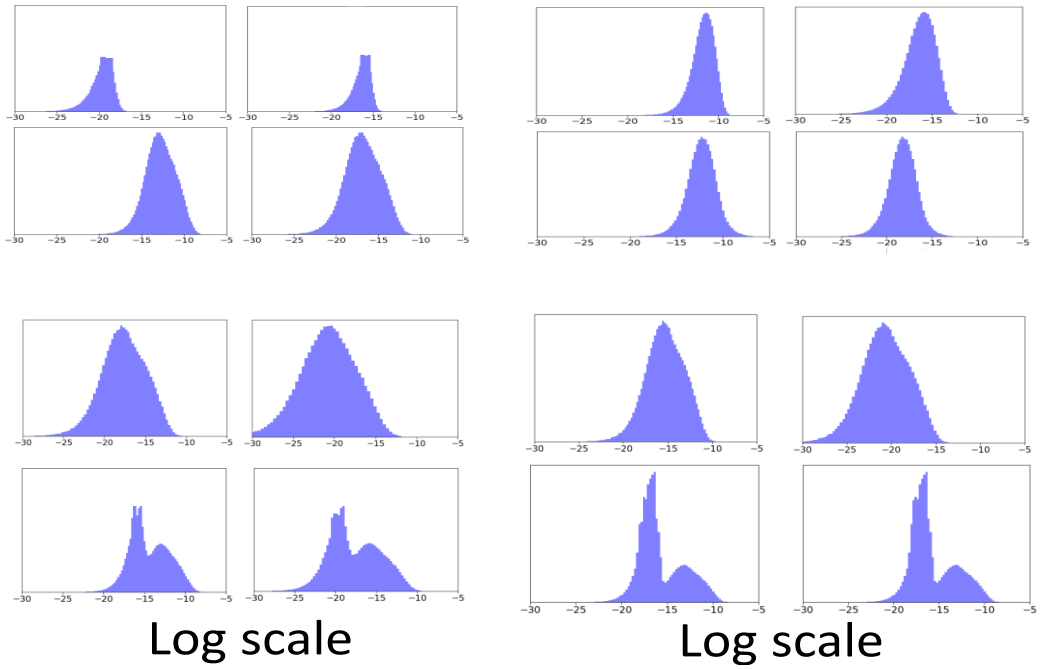


Figure A.4: **(Top left)** $\nabla\mathcal{A}_{i;1}$ distribution divided to the two components of $\nabla\mathcal{A}_{i;2}$ in two different layers. The left mode (top) originates from the zero element of $\nabla\mathcal{A}_{i;2}$ and the right mode (bottom) from the non zero elements, ResNet18 on ImageNet. **(Top right)** Histograms of neural gradients in different layers of Transformer (Vaswani et al., 2017) for WMT'16 En-De dataset in the decoder attention layer (top) and encoder embedding (bottom).**(Bottom left)** Histograms of neural gradients in different layers of DensNet121, ImageNet dataset. **(Bottom right)** Histograms of neural gradients in different layers of Vgg16, ImageNet dataset.

A.4 Floating point - relative error (Lognormal distribution)

We assume that $x \sim \text{Lognormal}(\mu, \sigma^2)$. Note that $E = \lfloor \ln x \rfloor \approx \ln x \sim \mathcal{N}(\mu, \sigma^2)$. We split the range into three parts according to E: (i) $-E_{\max} \leq E \leq E_{\max}$; (ii) $E \geq E_{\max}$; (iii) $E \leq -E_{\max}$, and calculate the expected contribution for each term of the relative error in Eq. (3):

A.4.1 The case of $-E_{\max} \leq E \leq E_{\max}$

In this case E can be expressed and no distortion appears due to clipping i.e., $E = E_q$. Therefore, the only contribution related to $\eta(n_1, n_2)$ is due to the distortion in the mantissa. This distortion is approximated through an additive noise $m_q = m + \varepsilon$, where the noise has a uniform distribution $\varepsilon \sim \mathbb{U}[-\Delta/2, \Delta/2]$. We can then calculate the expected relative error as follows

$$
\begin{aligned}
E\left[\left|\frac{x_q - x}{x}\right| \middle| -E_{\max} \leq E \leq E_{\max}\right] &= E\left[\left|\frac{m \cdot 2^E - m_q \cdot 2^{E_q}}{m \cdot 2^E}\right| \middle| -E_{\max} \leq E \leq E_{\max}\right] \\
&= E\left[\left|\frac{m \cdot 2^E - (m+\varepsilon) \cdot 2^E}{m \cdot 2^E}\right| \middle| -E_{\max} \leq E \leq E_{\max}\right] \\
&= E\left|\frac{\varepsilon}{m}\right| = E\left(\frac{|\varepsilon|}{|m|}\right) = E|\varepsilon| \cdot E\left|\frac{1}{m}\right|
\end{aligned}
$$

The last factorization is permissible since ε is independent of m. The expectation of $|\varepsilon|$ in the range $[-\Delta/2, \Delta/2]$ is calculated as follows:

$$E|\varepsilon| = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} \frac{|\varepsilon|}{\Delta} \cdot d\varepsilon = \frac{\varepsilon \cdot |\varepsilon|}{2\Delta}\Bigg|_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} = \frac{|\Delta|}{4} = \frac{1}{4 \cdot (2^{n_1} - 1)} \tag{A.6}$$

We turn to establish $E\left|\frac{1}{m}\right|$. Given a real value $x \in \mathbb{R}^+$, by definition m is positive and thus can be expressed as follows:

$$\left|\frac{1}{m}\right| = \frac{1}{m} = \frac{1}{2^{\ln x - \lfloor \ln x \rfloor}} = \frac{1}{2^{\varepsilon_m}} \tag{A.7}$$

where ε_m can be treated as a uniform variable in the range [0,1]. Therefore, we have that

$$E\left|\frac{1}{m}\right| = \int_0^1 \frac{1}{2^{\varepsilon_m}} \cdot d\varepsilon = -\frac{1}{\ln(2) \cdot 2^x}\Bigg|_0^1 = -\frac{1}{\ln(2) \cdot 2} \approx 0.721 \tag{A.8}$$

We can finally state the expected relative error for the case where x is supported by the dynamic range of x_q

$$E\left[\left|\frac{x_q - x}{x}\right| \Bigg| -E_{\max} \leq E \leq E_{\max}\right] = \frac{1}{8 \cdot \ln(2) \cdot (2^{n_1} - 1)} \tag{A.9}$$

We turn to find the probability $P(-E_{\max} \leq E \leq E_{\max})$. Assuming the lognormal distribution of x we get:

$$\begin{aligned} P(-E_{\max} \leq E \leq E_{\max}) &= \Phi\left(\frac{E_{\max}}{\sigma}\right) - \Phi\left(\frac{-E_{\max}}{\sigma}\right) \\ &= 2\Phi\left(\frac{E_{\max}}{\sigma}\right) - 1 \end{aligned} \tag{A.10}$$

where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution.

A.4.2 THE CASE OF $E \geq E_{\max}$

$$\begin{aligned} E\left[\left|\frac{x_q - x}{x}\right| \Bigg| E \geq E_{\max}\right] \cdot P(E \geq E_{\max}) = \\ &= \int_{E_{\max}}^{\infty} \frac{2^E - 2^{E_{\max}}}{2^E} \cdot \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{E^2}{2\sigma^2}} \mathrm{d}E \\ &= \frac{1}{2}\operatorname{erf}\left(\frac{E}{\sqrt{2}\sigma}\right) - 2^{E_{\max}-1}\mathrm{e}^{\frac{\sigma^2 \ln^2(4)}{8}} \operatorname{erf}\left(\frac{E}{\sqrt{2}\sigma} + \frac{\sigma \ln(4)}{\sqrt{8}}\right)\Big|_{E_{\max}}^{\infty} \\ &= 2^{E_{\max}-1}\mathrm{e}^{\frac{\sigma^2 \ln^2(2)}{2}} \left(\operatorname{erf}\left(\frac{\sigma \ln 2}{\sqrt{2}} + \frac{E_{\max}}{\sqrt{2}\sigma}\right) - 1\right) + \\ &- \frac{1}{2}\operatorname{erf}\left(\frac{E_{\max}}{\sqrt{2}\sigma}\right) + \frac{1}{2} \end{aligned} \tag{A.11}$$

A.4.3 THE CASE OF $E \leq -E_{\max}$

This is the underflow case. Here we have by definition that $x_q = 0$ and therefore we get

$$E\left[\left|\frac{x_q - x}{x}\right| \Bigg| E \leq -E_{\max}\right] = E\left[\left|\frac{0 - x}{x}\right| \Bigg| E \leq -E_{\max}\right] = 1 \tag{A.12}$$

This case has a probability of $P(E \leq -E_{\max}) = \Phi\left(-\frac{E_{\max}}{\sigma}\right) = 1 - \Phi\left(\frac{E_{\max}}{\sigma}\right)$

A.4.4 FINAL EXPECTED RELATIVE ERROR

Combining the three terms we have that

$$E\left[\left|\frac{x_q - x}{x}\right|\right] = \frac{2\Phi\left(\frac{E_{\max}}{\sigma}\right) - 1}{8 \cdot \ln(2) \cdot (2^{n_1} - 1)} + 2^{E_{\max}-1} e^{\frac{\sigma^2 \ln^2(2)}{2}} \left(\operatorname{erf}\left(\frac{\sigma \ln 2}{\sqrt{2}} + \frac{E_{\max}}{\sqrt{2}\sigma}\right) - 1\right) + - \frac{1}{2}\operatorname{erf}\left(\frac{E_{\max}}{\sqrt{2}\sigma}\right) + \frac{3}{2} - \Phi\left(\frac{E_{\max}}{\sigma}\right) \tag{A.13}$$

Given any N-bit FP format, seek a mantissa-exponent partition that minimizes the expected relative error such that $n_1 + n_2 = N - 1$. Minimizing Eq. (A.13) yields this optimal partition. To do so we set $n_1 = N - n_2 - 1$, equate the derivative to zero and solve. Empirically we found the computational cost of such solution is negligible and can be done online without affecting the NN running time.

Moreover, we can notice that for small values of σ a lognormal distributed tensor is similar to a normal distributed tensor. In such cases, the minimization of the relative error in Eq. (A.13) induced to allocate 0 bits to the exponent, i.e fixed point quantization. This fit to previous results (Banner et al., 2019; Baskin et al., 2018) which shows impressive results by quantization the fwd pass with fixed point quantization.

We can notice that although we use a different analytical measurement for stochastic pruning (cosine similarity) and for quantization (relative error), both of them are "analytical related" and define normalize values for the distance between original and noised tensor. The use of relative error in floating point format was already used in previous works[1].

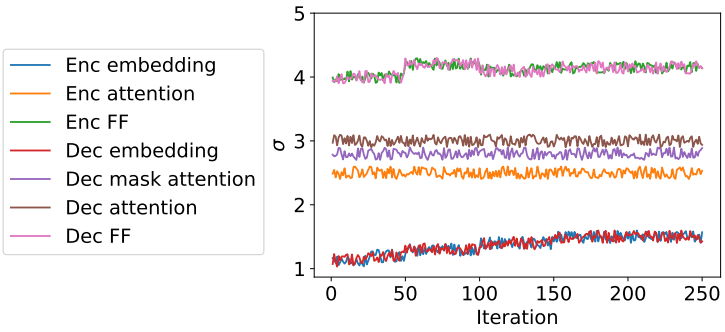


Figure A.5: (a) Std of layers gradient in different layers in the transformer (Vaswani et al., 2017) in WMT'16 En-De dataset assuming lognormal distribution. Notice the variability of std, which make the fp quantization challenging (Micikevicius et al., 2018).

A.5 FLOATING POINT - RELATIVE ERROR (NORMAL DISTRIBUTION)

We assume that $x \sim N(\mu, \sigma)$, and similar to Appendix A.4 we split the range into three parts according to E: (i) $-E_{\max} \leq E \leq E_{\max}$; (ii) $E \geq E_{\max}$; (iii) $E \leq -E_{\max}$. It is similar to the lognormal case, with maximum exponent of $2^{E_{max}}$.

A.5.1 THE CASE OF $-E_{\max} \leq E \leq E_{\max}$

$$\frac{1}{8 \cdot \ln(2) \cdot (2^{n_1} - 1)} 2\Phi\left(\frac{2^{E_{\max}}}{\sigma}\right) - 1 \tag{A.14}$$

[1] Accuracy and Stability of Numerical Algorithms: Second Edition; Nicholas J. Higham

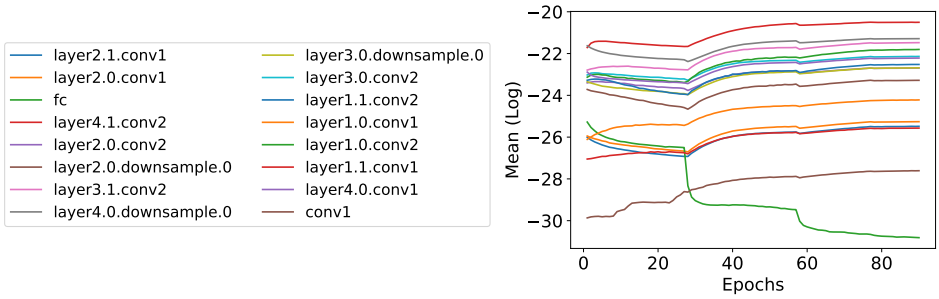


Figure A.6: Layer gradients means (log scale) in all layers of ResNet18, ImageNet dataset. Notice that the first ("Conv1") and last ("fc") suffer from a different statistic which requires a layerwise scaling factor.

A.5.2 THE CASE OF $E \geq E_{\max}$

$$2^{E_{\max}-1} e^{\frac{\sigma^2 \ln^2(2)}{2}} \left(\operatorname{erf}\left(\frac{\sigma \ln 2}{\sqrt{2}} + \frac{2^{E_{\max}}}{\sqrt{2}\sigma} \right) - 1 \right) - \frac{1}{2} \operatorname{erf}\left(\frac{2^{E_{\max}}}{\sqrt{2}\sigma} \right) + \frac{1}{2} \tag{A.15}$$

A.5.3 THE CASE OF $E \leq -E_{\max}$

$$1 - \Phi\left(\frac{2^{E_{\max}}}{\sigma} \right) \tag{A.16}$$

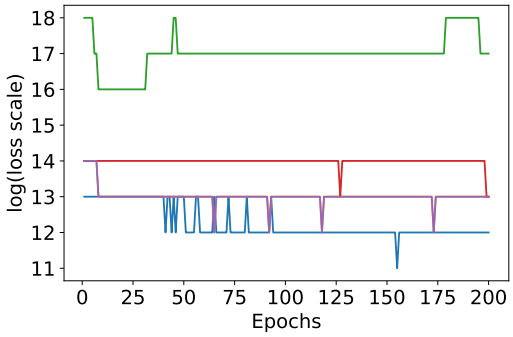


Figure A.7: The proposed gradient scale factor in different layer of ResNet18 in cifar10 dataset for FP5 format (1-4-0). Notice the variability across the layers which emphasize the importance of a different scale factor per layer.

A.6 GRADIENT SCALING

The suggested neural gradient scale algorithm presented in Algorithm 1 sacrifices the smallest elements to keep the largest gradients magnitude representable.

As shown in Fig. A.13b the maximum of the gradients' distribution at each layer is quite stable throughout the training. Therefore, we can infrequently sample the values of these maxima. This, in combination with the scaling in log scale, allows us to create an efficient hardware implementation of the suggested gradient scaling.

A.7 STOCHASTIC PRUNING COMPLEXITY

The suggested stochastic pruning method solution requires the mean and standard deviation of the input — each operation has a complexity of $\mathcal{O}(n)$ for an input size n. In practice, we perform this procedure of measuring the mean and standard deviation once every epoch. Additionally, it requires

Algorithm 1 Gradient scaling

1: **Input:** Neural gradient X, number of exponent bits n_2
2: **Output:** Quantized gradient X_Q using FP quantization Q_{FP}
3: $E_{\max} = 2^{n_2 - 1}$
4: $\mu_l = \lfloor \log_2 \max(|X|) \rfloor / \log_2 (E_{\max})$
5: $X_Q = 2^{\mu_l} \cdot Q_{FP}(X/2^{\mu_l})$

the solution of Eq. (8) which is independent in the input size. We empirically found it converges in a few iterations. In total, we found that the process of obtaining a threshold ,α, for a required sparsity adds less than 5% to the iteration time (for the single iteration, once in an epoch, when we calculate the threshold). For the rest of the iterations, there is no overhead caused by the stochastic pruning while achieving a significant improvement because of the induced sparsity.

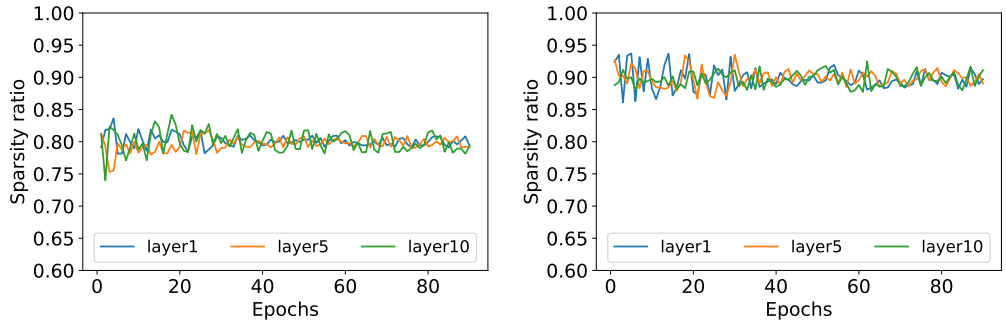
We can compare our method with top-k, which requires every iteration $\mathcal{O}(n \log n)$ for a naive implementation (sorting) and can be reduced using quick-select (Hoare, 1961) as was proposed by (Aamir Raihan & Aamodt, 2020) and vary between $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$.

A.8 Algorithm for bi-Modal lognormal pruning

Pruning of the bi-modal gradients ($\nabla \mathcal{A}_{i;1}$) is crucial for reducing the computational and bandwidth overhead. This is especially because it is involved in the calculation of the next layer gradient($\nabla \mathcal{A}_{i-1;3}$) and weights gradients of the convolution (∇W_i). Unlike $\nabla \mathcal{A}_{i;2}$, which naturally has a high sparsity level induced by the ReLU, $\nabla \mathcal{A}_{i;1}$ is naturally not sparse at all. In the following, we explain how to find the threshold α which induces the required sparsity ratio S in the bi-modal lognormal gradient.

To achieve even higher sparsity levels we can apply stochastic pruning method to the mixture of lognormally distributed gradients, $\nabla \mathcal{A}_{i;1}$. These gradients are comprised of two lognormal distributions with different means, variances, and the number of elements. Dealing with this distribution in the same theoretical framework as was done for the lognormal distribution is challenging. However, a simple solution can allow us to apply the same results to this distribution. We will rely on the fact that most of the values in the left mode are several orders of magnitude smaller than the ones in the right mode. Hence for a large enough sparsity value, we can say the following - if we will ignore the left mode and calculate the threshold assuming only the right mode exists then for almost all of the values in the left mode $|x| < \alpha \cdot \varepsilon$ and thus will be pruned. Therefore, to achieve an overall sparsity level of S we can find the threshold using the parameters of the right mode, adjusting the desired sparsity level down to account for the elements of the left mode that would be pruned as well.

In order to achieve a sparsity level S we use the following procedure — (1) calculate the ratio of the gradients that are in the left mode, denoted l, this is done by dividing the amount of values that were zero in the gradient before ($\nabla \mathcal{A}_{i;2}$) by the total number of elements in the gradient. (2) Here we assume that S is large enough, i.e. $l < S$. thus in order to achieve a total sparsity of S out of the entire tensor we need to sparsity $S' = \frac{S-l}{1-l}$ of the right mode. (3) Calculate the mean μ and variance σ of the right mode, by taking only the values in the tensor that their corresponding values in the pre-BN gradients are non-zero. (4) Solve the equation to find the threshold using σ and S' and apply stochastic pruning to the entire tensor using it.
We found that for high values of sparsity, above 0.7, the method is very accurate and achieves the desired sparsity thorough-out the training. For lower sparsity levels the actual sparsity induced tends to be a bit lower because the threshold is lower and the assumption that all of the values in the left mode are zeroed deteriorates progressively as the sparsity level decreases. In Fig. A.8 we show that the proposed method indeed achieves the desired sparsity.



(a) (b)

Figure A.8: The obtained sparsity in different layers of ResNet-18 with ImageNet dataset for required sparsity of 80% (a) and 90% (b)

A.9 Cosine similarity and heterogeneous stochastic pruning

A.9.1 Cosine similarity of stochastic pruning - general

Given a vector of gradients X, we would like to measure the cosine similarity (cosine of the angle) between the vector before and after stochastic pruning, i.e. X and $T_{\alpha,\varepsilon}(x)$:

$$\cos(\theta) = \frac{X \cdot T_{\alpha,\varepsilon}(x)}{||X||_2 \cdot ||T_{\alpha,\varepsilon}(x)||_2} \tag{A.17}$$

At high dimensions, the relative error made by substituting $||X||$ with $E||X||$ becomes asymptotically negligible (Biau & Mason, 2015). This distance concentration phenomenon simplifies Eq. A.17:

$$\cos(\theta) \approx \frac{\mathbb{E}[X \cdot T_{\alpha,\varepsilon}(X)]}{\mathbb{E}||X||_2 \cdot \mathbb{E}||T_{\alpha,\varepsilon}(x)||_2} \tag{A.18}$$

Solving the different components of eq. A.18 for $X \sim \text{lognormal}(0, \sigma^2)$ with a maximum value of $k\sigma$ (regarding the maximum value see Appendix A.9.4) with n elements, we get (see Appendix A.9.5):

$$\mathbb{E}[X \cdot T_{\alpha,\varepsilon}(X)] = \frac{1}{2}n \cdot e^{2\sigma^2}\left[1 + \text{erf}\left(\frac{k\sigma - 2\sigma^2}{\sigma\sqrt{2}}\right)\right] \tag{A.19}$$

$$\begin{aligned}\mathbb{E}||T_{\alpha,\varepsilon}(X)||_2^2 = &\frac{1}{2}n \cdot \alpha \cdot e^{\frac{\sigma^2}{2}}\left[1 - \text{erf}\left(\frac{\sigma^2 - \ln(\alpha)}{\sigma\sqrt{2}}\right)\right] \\ &+ \frac{1}{2}n \cdot e^{2\sigma^2}\left[\text{erf}\left(\frac{k\sigma - 2\sigma^2}{\sigma\sqrt{2}}\right) - \text{erf}\left(\frac{\ln(\alpha) - 2\sigma^2}{\sigma\sqrt{2}}\right)\right]\end{aligned} \tag{A.20}$$

$$\mathbb{E}||X||_2^2 = \frac{1}{2}n \cdot e^{2\sigma^2}\left[1 + \text{erf}\left(\frac{k\sigma - 2\sigma^2}{\sigma\sqrt{2}}\right)\right] \tag{A.21}$$

Fig. A.10 shows that these analytical results are in agreement with our simulations.
We notice in Fig. A.9a that the cosine similarity between original and pruned neural gradients can be used as a proxy for validation accuracy degradation under stochastic pruning. Interestingly, in Fig. A.9b we show that using the cosine similarity, we observed that stochastic pruning takes a different toll from the different layers i.e. pruning all layers to the same sparsity level damages some of them more than others. We propose an algorithm that using the analytical measured cosine similarity (Eq. (A.19),Eq. (A.20),Eq. (A.21)) preserves the cosine similarity of some of the layers, by decreasing their sparsity level, while increasing the sparsity level of other layers — maintaining the overall sparsity budget (mean sparsity of all the layers). The motivation for that and the algorithm are described in Appendix A.9.2 and Appendix A.9.3.

A.9.2 HETEROGENEOUS STOCHASTIC PRUNING

Using stochastic pruning, we noticed that different layers, with different distributional parameters, seem to be more sensitive to high levels of sparsity. This sensitivity can affect the overall accuracy achieved by the network. We notice two important parameters that contribute to the sensitivity of the layer: (1) its depth in the network, and (2) the error it suffers at a given sparsity level. The depth of the layer in the network is important because if a layer is deeper (closer to the output) then corrupting its gradients will cause the gradients reaching lower levels (closer to the input) to be corrupted as well. So, corruption of the gradients of the first layer, for example, affects only the first layer itself, while corrupting the gradients of the second layer affect both its gradients and the first layer's gradients, etc. To measure the distortion of the tensor due to the pruning, we used the cosine similarity (see definition in Eq. (A.17)) between the tensor before and after the stochastic pruning process. This measure reflects the preservation of the angle of the high-dimensional gradients tensor, under the pruning process. We noticed that in our homogeneous pruning scheme through all layers had the same desired sparsity level and a very similar achieved sparsity, the cosine similarity varied significantly between the layers. This is seen clearly in Fig. A.9b where each layer has a fairly stable cosine similarity that might be very different than the other layers, this is similar to the stability of each layer's distributional parameters that also vary between layers. In fact, as observed in Fig. A.18, the variance in cosine similarity between the layers can be explained exactly by the variance in the distributional parameters, for example the std σ.

A.9.3 HETEROGENEOUS ALGORITHM

While the effect of each layer's pruning on the overall outcome (accuracy) stems from its depth in the network and its distortion (measured through cosine similarity) its contribution to the overall sparsity is controlled by the number of its elements and its sparsity level. Since we aim to maximize the overall sparsity while preserving the baseline accuracy, we propose using different sparsity levels for the different layers. This would allow us to preserve the cosine similarity of the deeper layers, that affect the gradients of more layers while raising the sparsity of lower levels to maintain the overall sparsity. This can be done effectively because for most architectures the number of elements in the neural gradients of deeper layers is lower than these of shallow layers (usually because of max-pooling). This allows us to lower the sparsity of deeper layers significantly while raising the sparsity of shallower ones only slightly and still preserve the overall sparsity.

We propose an algorithm in which the user determines the overall desired sparsity and the minimum cosine similarities for the L' deepest layers, which can change from layer to layer. When determining each layer's threshold (at the beginning of each epoch, as in the regular homogeneous algorithm) if the layer is in the L' deepest layers the algorithm checks if its expected cosine similarity is greater than the minimum cosine similarity defined for that layer. If it is not, then the threshold is decreased so to achieve the minimum cosine similarity defined for that layer. After going through the L' deepest layers, that are first in the backward-pass, we know what was the sparsity for each layer so far. because we preserved the cosine similarity, the sparsity of these layers is expected to be lower than the required overall sparsity. In order to compensate for that, we can set a higher sparsity level for the rest of the layers, for which we don't define a minimum cosine similarity. This higher sparsity level can be calculated based on the sparsity of the first L' layers, which we already know, and the total number of elements in them compared to the rest of the layers.

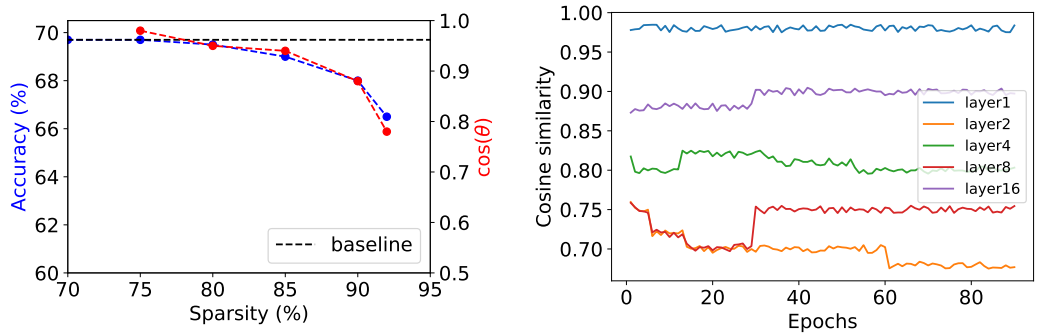
In order to refrain from over-pruning the shallow layers we set a maximum sparsity level for the layers for which we don't set a minimum cosine similarity level. From our experiments we find that setting a maximum of 0.96-0.98 will prevent over-pruning of these layers, which might deteriorate the final accuracy. Of course, using this maximum sparsity while aiming for both very high overall sparsity levels and high minimum cosine similarity for many of the layers might prevent us from reaching the desired overall sparsity. However, for most use-cases this doesn't pose significant restrictions on the parameters and at worse causes fairly small changes in overall sparsity.

Typical values we use to achieve high sparsity levels while preserving baseline accuracy are 0.95-0.98 minimum cosine similarity for the higher layers, where usually the deeper layers will receive higher minimum cosine similarities. For example, using ResNet18 on ImageNet, we have 4 basic blocks, we might preserve the last two with a minimum cosine similarity of 0.98 and the second one with 0.95, leaving the first block and the single layer before it, to compensate by having a higher sparsity level.

In Fig. A.19 we show an example of the different sparsity levels between the different layers when using heterogeneous pruning on ResNet18 trained on ImageNet.

A.9.4 NEURAL GRADIENTS LOGNORMAL DISTRIBUTIONS ARE TRUNCATED

When evaluating the cosine similarity of the different layers we noticed that the cosine similarity measure is very sensitive to the large valued components of the tensor, that is, pruning the same tensor after removing a few of the largest magnitude components will cause a far lower cosine similarity at the same sparsity level. This is because the cosine similarity represents the change in a high-dimensional angle, that is governed by the largest components.
Having first developed the equations in Appendix A.9 for a general lognormal distribution, we found a significant discrepancy between the expected and actual cosine similarities. That leads us to the understanding that the neural gradients' distributions actually differ slightly from the lognormal distribution by the lack of high-value components in the abundance that we might expect from tensors of their size where each value is drawn from a lognormal distribution. We verified and quantified this observation by examining the Gumbel distribution, that models the distribution of the maximum obtained from drawing a certain number of samples from a base distribution, in this case the lognormal distribution. We found that in fact the maximum values of the gradients were significantly lower than the ones expected from the Gumbel distribution of the corresponding lognormal distribution and size of the tensor. In Fig. A.22 we show examples of the actual maximum values of the tensors compared to the expected values using the Gumbel distribution.
Finally, we re-modeled the distribution of the gradients as a truncated lognormal distribution, that does not receive values above a certain value, denoted as k times the std σ. The value k is obtained from the distribution itself, like σ and μ. We found that taking the estimated k as the maximum of the obtained tensor makes a very noisy estimator, and does not match the actual cosine similarity closely, so in practice we take a high quantile (for example 0.997) of the distribution. It is important to note that this slight difference from the lognormal distribution is highly important when calculating the cosine similarity, but is negligible when dealing with sparsity, for example, because it is not sensitive to the lack of a few high-valued components.



(a) (b)

Figure A.9: (a) ResNet18 validation accuracy on the ImageNet data-set for different sparsity levels and the mean of the cosine similarity between the pruned and original gradient, notice the high correlation between the two. (b) The cosine similarity in different layers of ResNet18 on Imagenet dataset for 92 % pruning. Notice the variability between the different layers.

A.9.5 COSINE SIMILARITY OF STOCHASTIC PRUNING - FULL DERIVATION

In this section we extend on the full derivation of the cosine similarity for stochastic pruning.

To calculate the expected similarity between the n-dimensional tensors X and $T_{\alpha,\varepsilon}(X)$, we need to calculate $\mathbb{E}[X \cdot T(X)]$, $\mathbb{E}||X||_2$, and $\mathbb{E}||T_{\alpha,\varepsilon}(X)||_2$, as follows:

$$\mathbb{E}[X \cdot T_{\alpha,\varepsilon}(X)] = \mathbb{E}\left[\sum_i x_i \cdot T_{\alpha,\varepsilon}(x_i)\right] = \sum_i \mathbb{E}\left[x_i \cdot T_{\alpha,\varepsilon}(x_i)\right] = n \cdot \mathbb{E}\left[x \cdot T_{\alpha,\varepsilon}(x)\right] \quad \text{(A.22)}$$

And similary:

$$\mathbb{E}||X||_2 = n \cdot \mathbb{E}\left[x\right]^2 \tag{A.23}$$

$$\mathbb{E}||T_{\alpha,\varepsilon}\left(X\right)||_2 = n \cdot \mathbb{E}\left[T_{\alpha,\varepsilon}\left(x\right)\right]^2 \tag{A.24}$$

Assuming $X \sim \text{LogNormal}(0, \sigma^2)$ with maximum value $k\sigma$, the cosine similarity between X and $T_{\alpha,\varepsilon}\left(X\right)$ is :

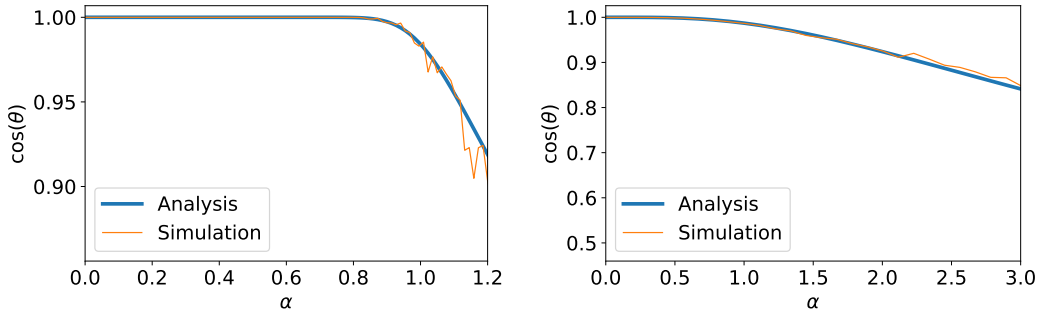
$$\mathbb{E}||X||_2^2 = \frac{1}{2}n \cdot e^{2\sigma^2}\left[1 + \text{erf}\left(\frac{\ln(k\sigma) - 2\sigma^2}{\sigma\sqrt{2}}\right)\right] \tag{A.25}$$

$$\begin{aligned}
\mathbb{E}||T(X)||_2^2 &= n\int_0^1\int_{\varepsilon\alpha}^{\alpha} \alpha^2 \cdot f(x)\,dx\,d\varepsilon + n\int_{\alpha}^{k\sigma} x^2 \cdot f(x)\,dx \\
&= n\int_0^1\int_{\varepsilon\alpha}^{\alpha} \frac{\alpha^2}{x\sigma\sqrt{2\pi}} e^{-\frac{\ln(x)^2}{2\sigma^2}}\,dx\,d\varepsilon + n\int_{\alpha}^{k\sigma} \frac{x}{\sigma\sqrt{2\pi}} e^{-\frac{\ln(x)^2}{2\sigma^2}}\,dx \\
&= \frac{1}{2}n \cdot \alpha^2 \int_0^1 \left[\text{erf}\left(\frac{\ln(\alpha)}{\sigma\sqrt{2}}\right) - \text{erf}\left(\frac{\ln(\varepsilon\alpha)}{\sigma\sqrt{2}}\right)\right] d\varepsilon \\
&+ \frac{1}{2}n \cdot e^{2\sigma^2}\left[\text{erf}\left(\frac{\ln(k\sigma) - 2\sigma^2}{\sigma\sqrt{2}}\right) - \text{erf}\left(\frac{\ln(\alpha) - 2\sigma^2}{\sigma\sqrt{2}}\right)\right] \\
&= \frac{1}{2}n \cdot \alpha^2 \cdot \text{erf}\left(\frac{\ln(\alpha)}{\sigma\sqrt{2}}\right) \\
&- \frac{1}{2}n \cdot \left[\alpha \cdot e^{\frac{\sigma^2}{2}} \text{erf}\left(\frac{\sigma^2 - \ln(\varepsilon\alpha)}{\sigma\sqrt{2}}\right) + \alpha^2\varepsilon \cdot \text{erf}\left(\frac{\ln(\varepsilon\alpha)}{\sigma\sqrt{2}}\right)\right]\Big|_0^1 \\
&+ \frac{1}{2}n \cdot e^{2\sigma^2}\left[\text{erf}\left(\frac{\ln(k\sigma) - 2\sigma^2}{\sigma\sqrt{2}}\right) - \text{erf}\left(\frac{\ln(\alpha) - 2\sigma^2}{\sigma\sqrt{2}}\right)\right] \\
&= \frac{1}{2}n \cdot \alpha^2 \cdot \text{erf}\left(\frac{\ln(\alpha)}{\sigma\sqrt{2}}\right) \\
&- \frac{1}{2}n \cdot \left[\alpha \cdot e^{\frac{\sigma^2}{2}} \text{erf}\left(\frac{\sigma^2 - \ln(\alpha)}{\sigma\sqrt{2}}\right) + \alpha^2 \cdot \text{erf}\left(\frac{\ln(\alpha)}{\sigma\sqrt{2}}\right) - \alpha \cdot e^{\frac{\sigma^2}{2}}\right] \\
&+ \frac{1}{2}n \cdot e^{2\sigma^2}\left[\text{erf}\left(\frac{\ln(k\sigma) - 2\sigma^2}{\sigma\sqrt{2}}\right) - \text{erf}\left(\frac{\ln(\alpha) - 2\sigma^2}{\sigma\sqrt{2}}\right)\right] \\
&= \frac{1}{2}n \cdot \alpha \cdot e^{\frac{\sigma^2}{2}}\left[1 - \text{erf}\left(\frac{\sigma^2 - \ln(\alpha)}{\sigma\sqrt{2}}\right)\right] \\
&+ \frac{1}{2}n \cdot e^{2\sigma^2}\left[\text{erf}\left(\frac{\ln(k\sigma) - 2\sigma^2}{\sigma\sqrt{2}}\right) - \text{erf}\left(\frac{\ln(\alpha) - 2\sigma^2}{\sigma\sqrt{2}}\right)\right]
\end{aligned} \tag{A.26}$$

$$
\begin{aligned}
\mathbb{E}[X \cdot T(X)] &= n \int_0^1 \int_{\varepsilon\alpha}^{\alpha} \alpha \cdot x \cdot f(x)\, dx\, d\varepsilon \cdot + n \int_{\alpha}^{k\sigma} x^2 \cdot f(x)\, dx \\
&= \int_0^1 \int_{\varepsilon\alpha}^{\alpha} \frac{\alpha}{\sigma\sqrt{2\pi}} e^{-\frac{\ln(x)^2}{2\sigma^2}}\, dx\, d\varepsilon + n \int_{\alpha}^{k\sigma} \frac{x}{\sigma\sqrt{2\pi}} e^{-\frac{\ln(x)^2}{2\sigma^2}}\, dx \\
&\underset{y=\ln(x)}{=} n \int_0^1 \int_{\ln(\varepsilon\alpha)}^{\ln(\alpha)} \frac{\alpha}{\sigma\sqrt{2\pi}} e^{-\frac{y^2}{2\sigma^2}} e^{y}\, dy\, d\varepsilon + n \int_{\ln(\alpha)}^{\ln(k\sigma)} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{y^2}{2\sigma^2}} e^{2y}\, dy \\
&= n \int_0^1 \int_{\ln(\varepsilon\alpha)}^{\ln(\alpha)} \frac{\alpha}{\sigma\sqrt{2\pi}} e^{-\frac{(y-\sigma^2)^2}{2\sigma^2}} e^{\frac{\sigma^2}{2}}\, dy\, d\varepsilon + n \int_{\ln(\alpha)}^{\ln(k\sigma)} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-2\sigma^2)^2}{2\sigma^2}} e^{2\sigma^2}\, dy \\
&= n \cdot \alpha \cdot e^{\frac{\sigma^2}{2}} \cdot \frac{1}{2} \int_0^1 \left(1 + \operatorname{erf}\left(\frac{y-\sigma^2}{\sigma\sqrt{2}}\right)\right) \Big|_{\ln(\varepsilon\alpha)}^{\ln(\alpha)} d\varepsilon \\
&+ n \cdot e^{2\sigma^2} \cdot \frac{1}{2} \left(1 + \operatorname{erf}\left(\frac{y-2\sigma^2}{\sigma\sqrt{2}}\right)\right) \Big|_{\ln(\alpha)}^{\ln(k\sigma)} \\
&= \frac{1}{2} n \cdot e^{\frac{\sigma^2}{2}} \cdot \alpha \int_0^1 \left[\operatorname{erf}\left(\frac{\ln(\alpha)-\sigma^2}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{\ln(\varepsilon\alpha)-\sigma^2}{\sigma\sqrt{2}}\right)\right] d\varepsilon \\
&+ \frac{1}{2} n \cdot e^{2\sigma^2} \left[\operatorname{erf}\left(\frac{\ln(k\sigma)-2\sigma^2}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{\ln(\alpha)-2\sigma^2}{\sigma\sqrt{2}}\right)\right] \\
&= \frac{1}{2} n \cdot e^{\frac{\sigma^2}{2}} \cdot \alpha \cdot \operatorname{erf}\left(\frac{\ln(\alpha)-\sigma^2}{\sigma\sqrt{2}}\right) \\
&- \frac{1}{2} n \cdot e^{\frac{\sigma^2}{2}} \left[e^{\frac{3\sigma^2}{2}} \operatorname{erf}\left(\frac{2\sigma^2 - \ln(\alpha\varepsilon)}{\sqrt{2}\sigma}\right) - \alpha\varepsilon \cdot \operatorname{erf}\left(\frac{\sigma^2 - \ln(\varepsilon\alpha)}{\sqrt{2}\sigma}\right)\right] \Big|_0^1 \\
&+ \frac{1}{2} n \cdot e^{2\sigma^2} \left[\operatorname{erf}\left(\frac{\ln(k\sigma)-2\sigma^2}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{\ln(\alpha)-2\sigma^2}{\sigma\sqrt{2}}\right)\right] \\
&= \frac{1}{2} n \cdot e^{\frac{\sigma^2}{2}} \cdot \alpha \cdot \operatorname{erf}\left(\frac{\ln(\alpha)-\sigma^2}{\sigma\sqrt{2}}\right) \\
&- \frac{1}{2} n \cdot e^{\frac{\sigma^2}{2}} \left[e^{\frac{3\sigma^2}{2}} \operatorname{erf}\left(\frac{2\sigma^2 - \ln(\alpha)}{\sqrt{2}\sigma}\right) - \alpha \cdot \operatorname{erf}\left(\frac{\sigma^2 - \ln(\alpha)}{\sqrt{2}\sigma}\right) - e^{\frac{3\sigma^2}{2}}\right] \\
&+ \frac{1}{2} n \cdot e^{2\sigma^2} \left[\operatorname{erf}\left(\frac{\ln(k\sigma)-2\sigma^2}{\sigma\sqrt{2}}\right) - \operatorname{erf}\left(\frac{\ln(\alpha)-2\sigma^2}{\sigma\sqrt{2}}\right)\right] \\
&= \frac{1}{2} n \cdot e^{2\sigma^2} \left[1 - \operatorname{erf}\left(\frac{2\sigma^2 - \ln(\alpha)}{\sqrt{2}\sigma}\right)\right] \\
&+ \frac{1}{2} n \cdot e^{2\sigma^2} \left[1 - \operatorname{erf}\left(\frac{\ln(\alpha)-2\sigma^2}{\sigma\sqrt{2}}\right)\right] \\
&= \frac{1}{2} n \cdot e^{2\sigma^2} \left[1 + \operatorname{erf}\left(\frac{\ln(k\sigma)-2\sigma^2}{\sigma\sqrt{2}}\right)\right]
\end{aligned}
\tag{A.27}
$$

A.10 ENCODING

We suggest a custom encoding method that fully utilizes the abundance of both zeros and $\pm\alpha$ generated by the stochastic pruning, at their relative frequency. As shown in Fig. A.11a the compression ratio achieved by the proposed encoding, relative to original FP32, is equivalent to quantizing to 4 bits at 80% sparsity, and to only 2 bits at 90%. In Fig. A.11b we show the actual bits per value for the results shown in Section 6 using the proposed encoding.



(a) (b)

Figure A.10: Comparison between simulation and analysis (Eq. (A.19), Eq. (A.20) and Eq. (A.21)) of the cosine similarity for $\sigma = 3$ (a) and $\sigma = 5$(b).

Table A.4: Simple proposed encoding for the values after stochastic pruning

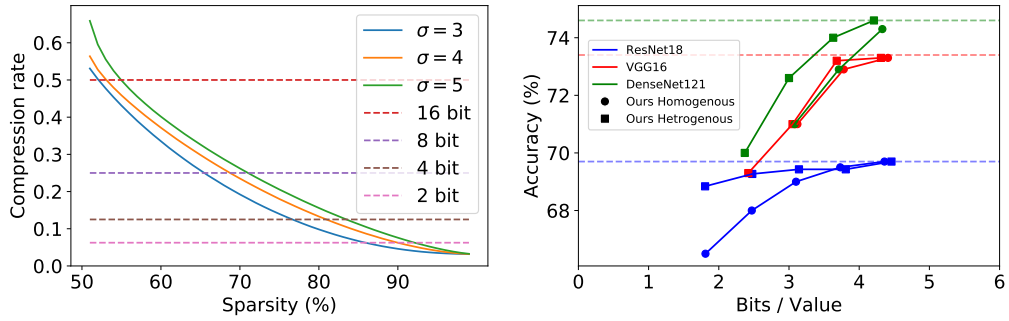
Value	encoding
0	0
α	100
$-\alpha$	101
starting a FP value	11

A.11 LAYER GRADIENTS STABILITY

The measure of mean and std in log scale of the layer gradients in ResNet18, cifar100 dataset is shown in Fig. A.12. Notice the stability inside each epoch, that allows us to sample them once in an epoch to calculate the corresponding threshold.

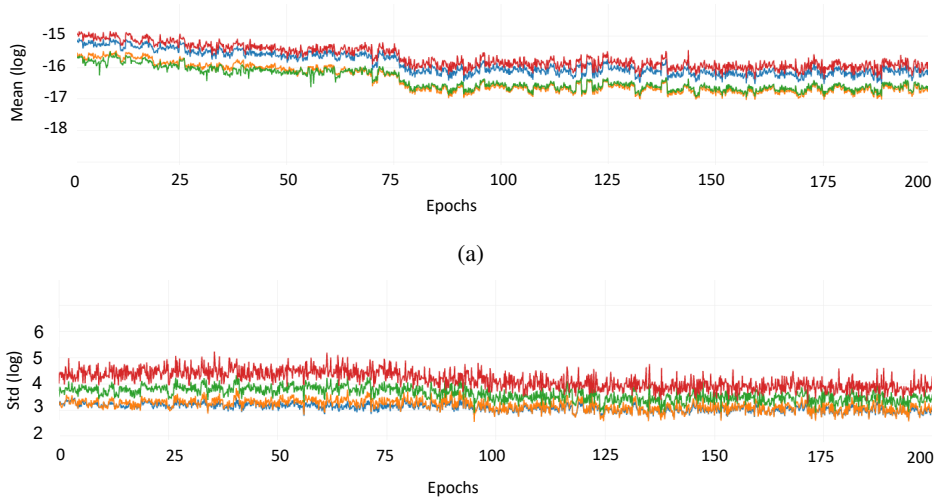
A.12 EXPERIMENTS

We implement all the experiments in PyTorch , models are trained on ImageNet dataset (Deng et al., 2009) and are evaluated on three different architectures ResNet18 (He et al., 2016), DenseNet121 (Huang et al., 2017) and VGG16 (Simonyan & Zisserman, 2015). We train the models with the standard regime for 90 epochs, using SGD with momentum, reducing the learning rate by a factor of $\frac{1}{10}$ after 30 and 60 epochs. Note the difference between the ATP and our homogeneous method is only the way to find the threshold α for a required sparsity. Therefore, we believe the difference in the accuracy for the same sparsity (we had better results) is probably the result of implementation issues. Our heterogeneous method further improves the validation accuracy for a given sparsity level, as shown in Fig. 5. Lastly, we note that ATP (Ye et al., 2019) for some reason only reported training accuracy, but not to be too strict (it might have been just a typo) we assumed that their validation accuracy was the same as the training accuracy — as was in our experiments.



(a) (b)

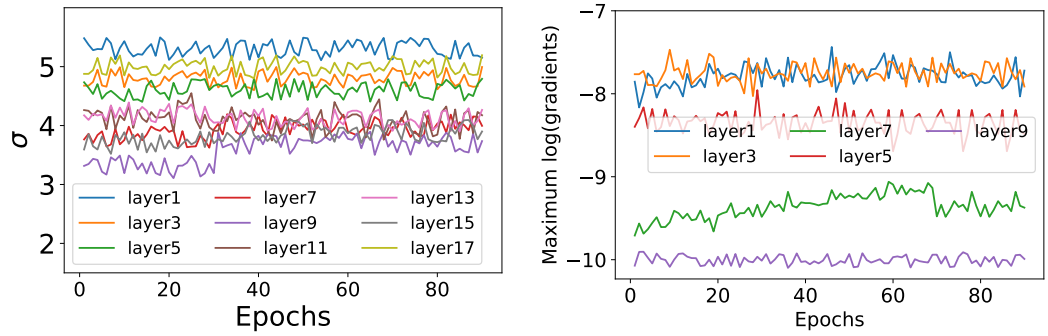
Figure A.11: (a) Compression rate achieved for the proposed encoding for a synthetic lognormal distributed tensor for common gradients σ and the comparison to quantization to different bitwidth. Notice for 90% sparsity the compression rate is similar to quantization to 2 bits. (b) Bits for each value using the proposed encoding method for the results in Fig. 5.



(a)

(b)

Figure A.12: The gradients mean (a) and std (b) of ResNet-18 , Cifar100 dataset in different layers. Notice they are stable across each epoch - so there is the posibility to calculate the threshold for the required sparsity once in an epoch



(a) (b)

Figure A.13: **(a):** σ of different layers throughout training (ResNet18 on ImageNet). **(b):** Maximum of the neural gradient at log scale throughout training (ResNet18 on ImageNet).

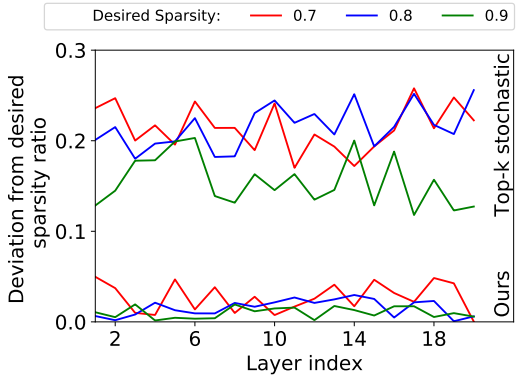
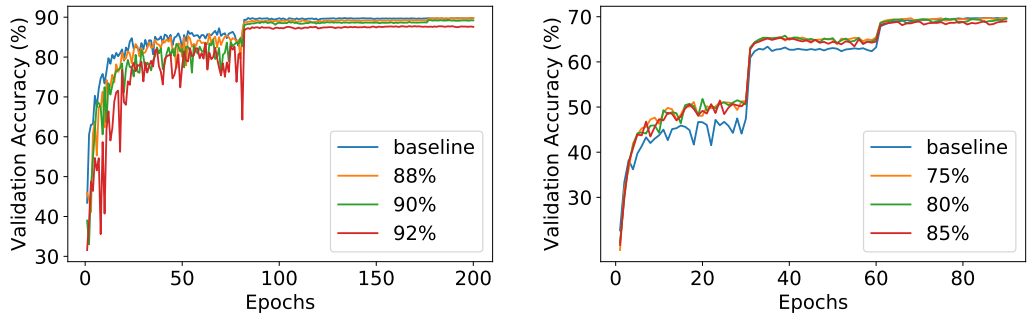


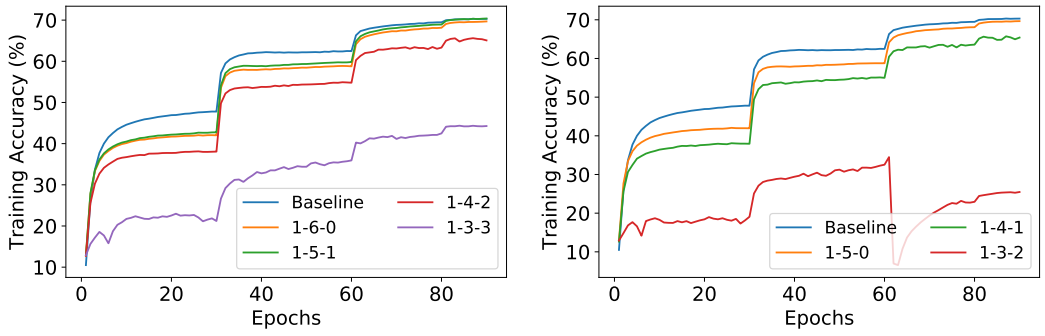
Figure A.14

Figure A.15: Comparison of the deviation from the desired sparsity ratio of stochastic pruning Vs "top-k" + stochastic pruning, i.e finding the threshold α with "top-k" and then applying stochastic pruning. (Eq. (5)) in different layers. Notice the high deviation in "top-k" + stochastic pruning is uniform across the layers



(a) (b)

Figure A.16: ResNet18 training convergence with differente sparsity ratios for Cifar10(a) and ImagenNet(b) datasets.



(a) (b)

Figure A.17: FP7(a) and FP6(b) training convergence in different formats in ResNet18, ImageNet dataset. Notice the results fit the analysis in Fig. 3b

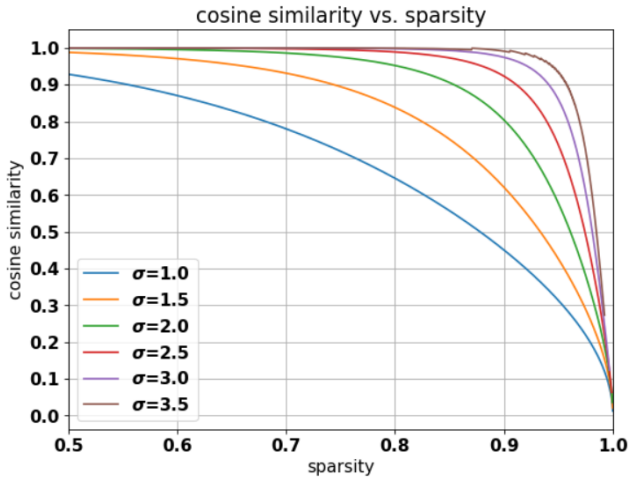


Figure A.18: Cosine similarity vs. sparsity for different σ. For all distributions plotted here $k = 2.5$. The values were calculated analytically using our analyses of the sparsity and cosine similarity using stochastic pruning. Notice that for the same sparsity values different distributions vary significantly in cosine similarity, this explains the difference in cosine similarity of the different layers when using homogeneous stochastic pruning (all layers having the same sparsity level).

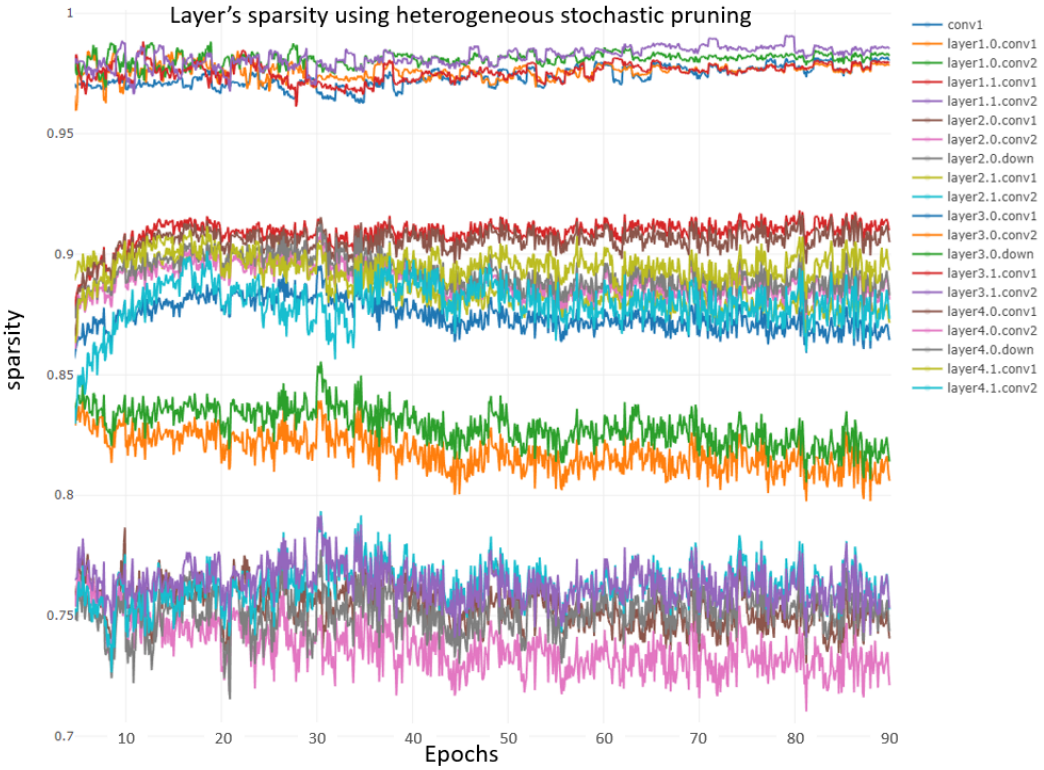


Figure A.19: Actual sparsity of the different layers of ResNet18 trained on ImageNet using heterogeneous stochastic pruning. The desired sparsity was 92% and the overall achieved one is 91.5%, the minimum cosine similarity for blocks 3 & 4 was 0.98 and for block 2 was 0.95, and the maximum allowed sparsity for the rest of the layers (block 1 and "conv1" before the first block) was 0.98. Notice that the high sparsity levels were induced on the first block and "conv1", and that some of the layers, especially from block 2, had much lower sparsity levels in order to preserve its cosine similarity. For this experiment the validation accuracy was less than 1% degradation from baseline, compared to over 3% degradation for the homogeneous algorithm at similar sparsity level.

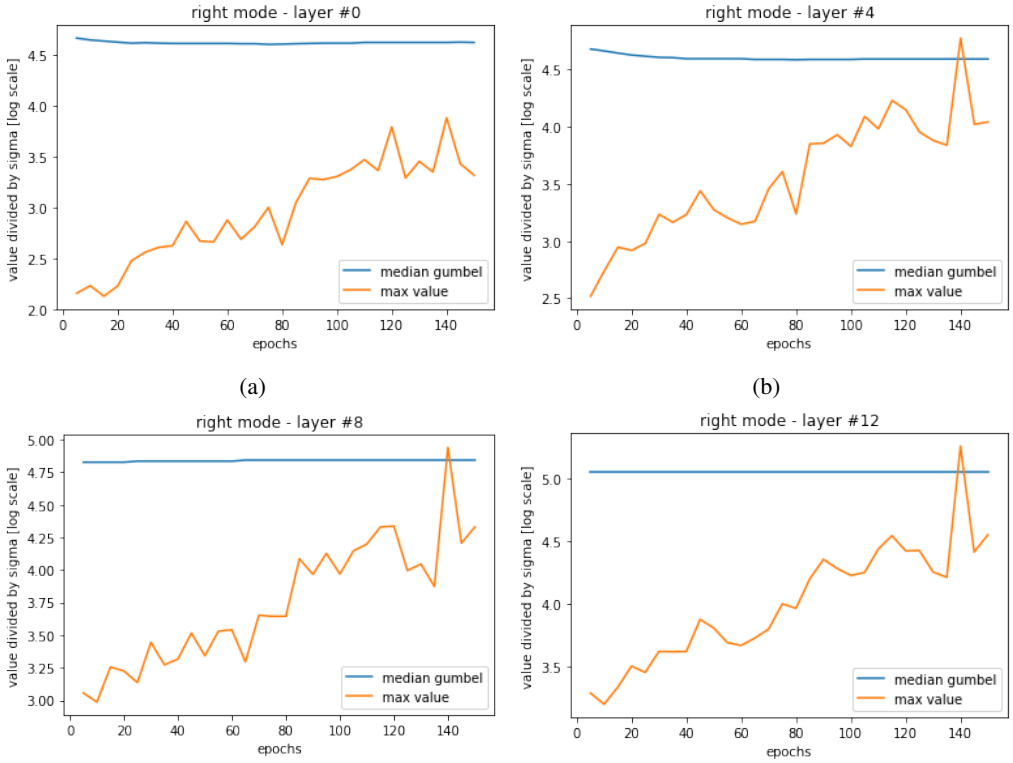


Figure A.20: Comparison between the median value of the Gumbel distribution and the actual maximum value of the gradients for ResNet18 trained on Cifar10. The Gumbel distribution represents the distribution of the maximum value for a tensor of the size of the gradients tensor (each layer of different size) drawn from a lognormal distribution with the mean and std of the tensor. Here we use only the right mode of the bi-modal mixture of lognormal distributions. Notice that the actual maximum value is orders of magnitudes lower than the expected maximum value from the Gumbel distribution (y-axis is in log scale, and the value is divided by the std). The actual tensors are truncated at levels of approximately 3-5 σ. This observation lead us to model the distribution as a truncated lognormal distribution when we derived the cosine similarity analytically, because the cosine similarity is more sensitive to the presence of high-valued components in the tensor. The numbering of the layers here is from the deepest to the shallowest, according to the order of the gradients flow in the backward-pass. Notice that layer 12 gradients has twice as many elements as layer 8 that has twice as many as layers 4 and 0, thus the median of their Gumbel distributions is higher, measuring in sigmas, because if we draw more values we expect the maximum value to be larger.

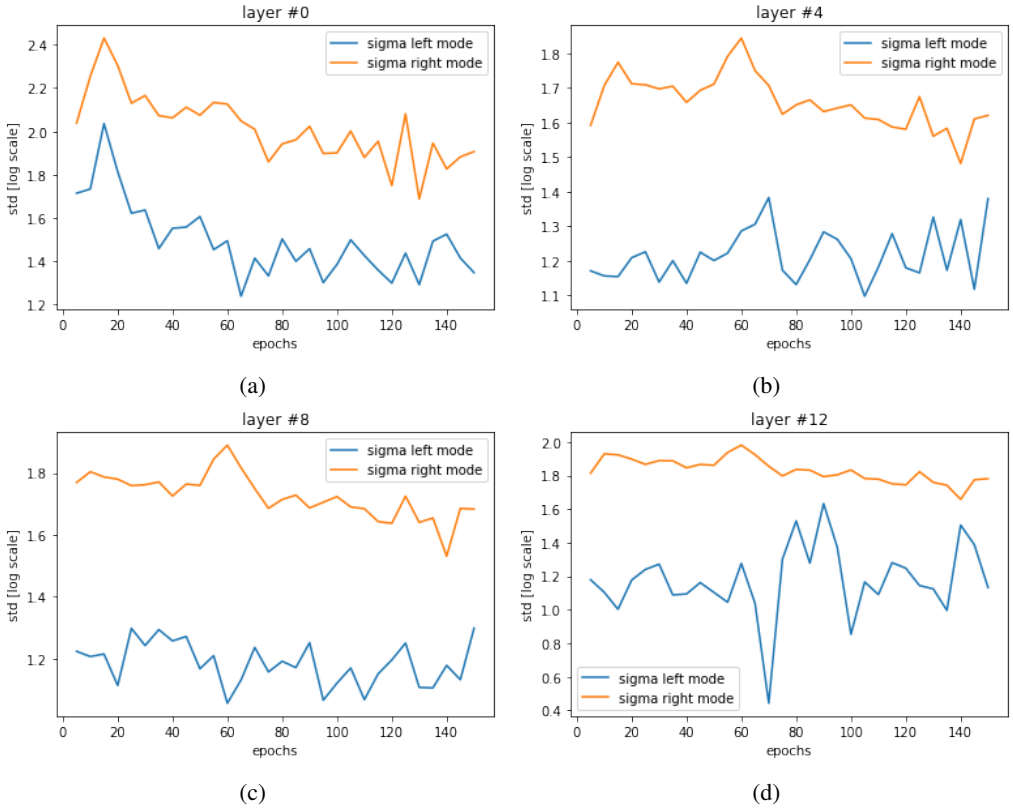


Figure A.21: The std (σ) of each of the two modes of the gradients for ResNet18 trained on Cifar10, on various layers. Notice that the left mode is of lower variance. The numbering of the layers here is from the deepest to the shallowest, according to the order of the gradients flow in the backward-pass.

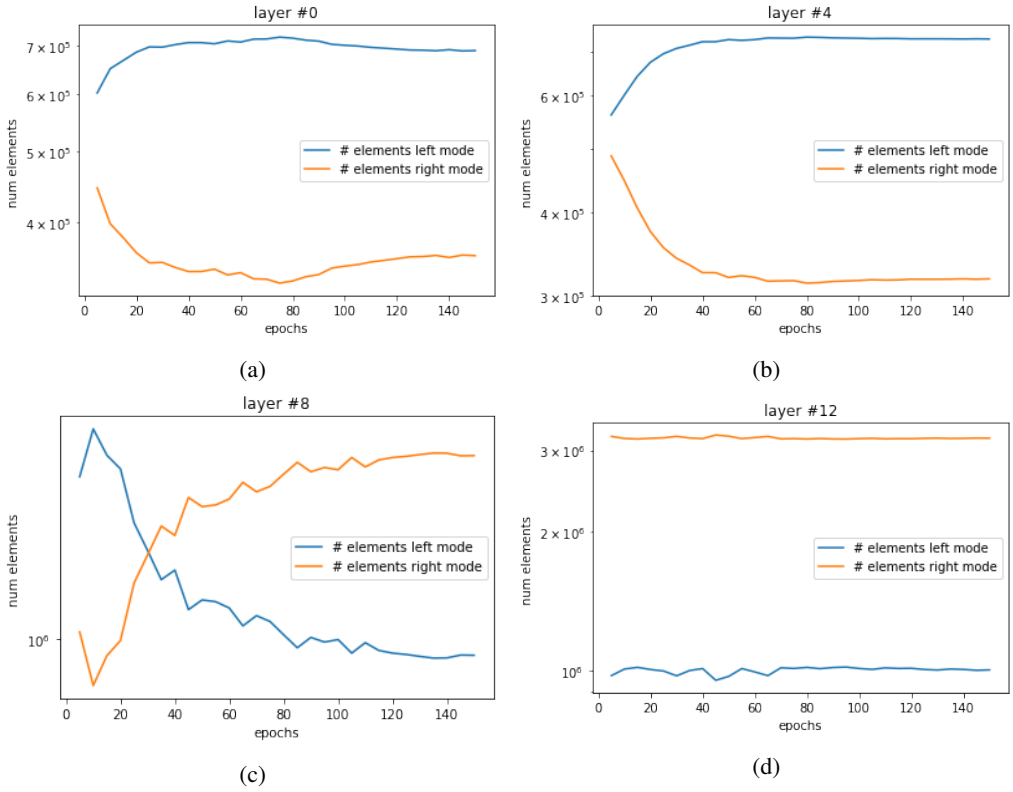


Figure A.22: The number of elements of each of the two modes of the gradients for ResNet18 trained on Cifar10, on various layers. Notice that the y-axis is in log scale. Different layers vary in behaviour, but the dominant effect is the increase in the proportion of the left mode, indicating that a large portion of the values are mapped to zero by the consequent ReLU layer, because the left modes' values originate from zeros values in the ReLU gradients. Up to 80% of values might be in the left mode. This has major consequences when trying to prune the bi-modal distributions to values lower than that using our algorithm. However this is less of a problem when pruning to high sparsity levels. The numbering of the layers here is from the deepest to the shallowest, according to the order of the gradients flow in the backward-pass.

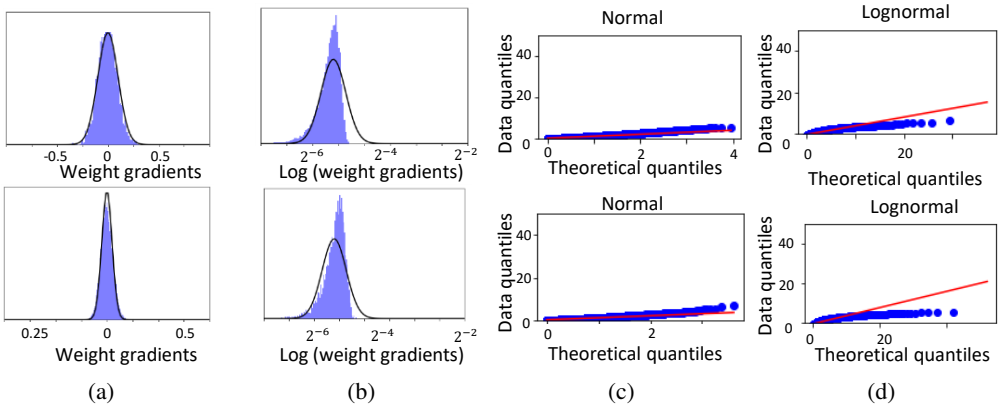


Figure A.23: Identifying the distribution of weight gradients (normal vs. lognormal) for additional layers in ResNet18 - ImageNet dataset, similar to Fig. 2. We see the weight gradients are much closer to a normal distribution than a lognormal distribution.

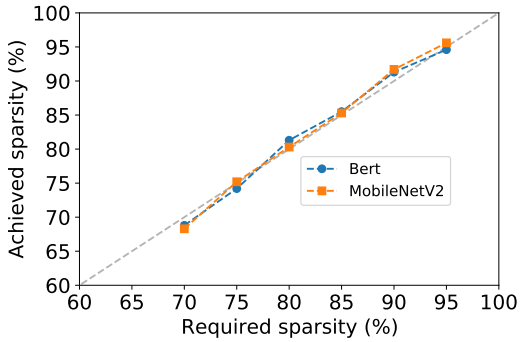


Figure A.24: Comparison of the required and achieved sparsity of our method for MobileNetV2 and Bert models (Similar to Fig. 5).