# **Reducing Storage of Pretrained Neural Networks by Rate-Constrained Quantization and Entropy Coding**

Alexander Conzelmann Department of Computer Science University of Tübingen Tübingen, Germany a.conzelmann@uni-tuebingen.de Robert Bamler Department of Computer Science University of Tübingen Tübingen, Germany robert.bamler@uni-tuebingen.de

# Abstract

The ever-growing size of neural networks poses serious challenges on resourceconstrained devices, such as embedded sensors. Compression algorithms that reduce their size can mitigate these problems, provided that model performance stays close to the original. We propose a novel post-training compression framework that combines rate-aware quantization with entropy coding by (1) extending the well-known layer-wise loss by a quadratic rate estimation, and (2) providing locally exact solutions to this modified objective following the Optimal Brain Surgeon (OBS) method. Our method allows for very fast decoding and is compatible with arbitrary quantization grids. We verify our results empirically by testing on various computer-vision networks, achieving a 20-40% decrease in bit rate at the same performance as the popular compression algorithm NNCodec.

Our code is available at https://github.com/Conzel/cerwu.

# 1 Introduction

While neural networks have achieved impressive results on various tasks, their large computational demands require many neural network applications to rely on server-side deployment. This increases latency and can cause concerns for end users regarding privacy and regulatory constraints [49, 27]. Thus, there has been an increased effort to move machine learning pipelines back onto user devices, which usually involves techniques that reduce their resource demands [27]. The works on neural network compression can be roughly divided into two groups: (1) works that aim to reduce the *server-side* costs of training [37, 24], fine-tuning [12, 47] or inference [36, 28, 44]. These methods often save computational costs or reduce GPU memory requirements. And (2) works that focus on *embedded and edge devices*, aiming to produce extremely small models still suitable for inference [29, 22], to reduce energy demands [35, 5] or to decrease the storage requirements [9, 6]. While both groups use similar techniques, such as pruning, quantization and entropy coding, the methods might not be used interchangeably, as they target different architectures and scenarios.

Our work explicitly focuses on the storage requirements of neural networks, which we argue is important for more widespread adoption of neural networks on edge devices. These devices often have limited storage capacity or network bandwidth for software updates, causing model-size issues with even the simplest of neural networks (even ResNets [26] have an uncompressed storage size of hundreds of megabytes). The importance of storage size has been reinforced with the introduction of the ISO/IEC 15938-17 standard, a technical standard for the compression of neural networks.

So far, few other works explicitly target the storage size of neural networks. When focusing just on inference speed, quantization (i.e., saving each parameter with a lower precision) is often enough, as current GPU kernels only experience a speed-up for 4-bit or 8-bit quantization (see [43]). However, to reduce storage size even further, a more powerful technique is to use entropy coding, which builds

Preprint. Under review.

a probabilistic model of the data and then encodes the data points into bit strings whose length is proportional to their *information content* (negative log-probability), allowing us to encode parameters with "fractional bits", in some cases even achieving rates below 1 bit per weight.

In this work, we combine quantization with entropy coding, and propose a novel approach that produces highly compressible quantized layer representations by adding a quadratic rate-estimation to the layer-wise loss [42], searching for rate-distortion optimal quantization choices, and then providing entropy-regularized optimal weight updates following the Optimal Brain Surgeon (OBS) [25] framework. In summary, our proposed network compression scheme

- achieves extremely low storage cost for the compressed networks;
- allows for extremely fast decoding of compressed networks;
- produces quantized representations suitable for fast inference using integer arithmetic; and
- is flexible in regards to the choice of quantization grid and entropy model used, allowing users to, e.g., trade off decompression or inference speed against model performance.

We name our method **CERWU** (Compression with Entropy-Regularized Weight Updates). In the rest of this paper, we discuss related work (section 2) and the information-theoretical background (section 3), derive the proposed entropy-regularized weight updates (section 4), and verify the effectiveness of our method empirically (section 5) on various networks from the computer vision community. We conclude with a discussion of limitations of our method (section 6).

# 2 Related Work

Various target scenarios exist in network compression. Our method is a *post-training* compression method that reduces the *storage size* of a network. Other compression schemes focus instead on decreasing the memory footprint on GPUs [51, 36], reducing energy costs [33, 8, 35], increasing inference speed [38, 11] or they require re-training for each tested compression setting [5, 7].

Most related work focuses on network *quantization* rather than compression (often aiming to improve inference speed). We discuss the difference between the two in section 3.1 and existing quantization methods in section 3.3. There are fewer existing works with our focus on storage size. Han et al. [23] combine pruning, quantization, weight-sharing and entropy coding to achieve very high compression factors. Universal Neural Network Compression [9] uses universal quantization [58] together with vector quantization. Wiedemann et al. [56] propose to use an entropy-constraint together with a sparsification process, resulting in parameter representations that allow for high compression ratios.

The *Neural Network Compression and Representation Standard* (ISO/IEC 15938-17) [32], with its reference implementation NNCodec [6], defines a compression pipeline encompassing parameter reduction (pruning, sparsification, parameter sharing, et cetera), quantization, and entropy coding, as well as interoperability with well-known neural network exchange formats such as ONNX [3].

### **3** Background

### 3.1 Compression vs. Quantization

Neural network compression is often conflated with quantization, but the two are fundamentally different. In its most basic form, quantization reduces the precision of each network weight to a fixed (integer) bit-width (e.g., 4 or 8 bits per weight). By contrast, compression is a more global operation that aims to find the most compact binary representation of a network by packing the most relevant information contained in all weights into a bit string of shortest possible length (called *bit rate*). Compression thus effectively maps weights to varying bit-widths, and modern compression methods [48, 45, 41, 16, 4] are not even constrained to bit boundaries, i.e., they can effectively assign fractional bit-widths to weights, including widths below 1 bit per weight, see section 3.2 below.

The simplest way to combine quantization and compression is to first quantize the weights to a fixed bit-width and then apply lossless compression. However, this is empirically far from optimal (see ablation "CERWU- $\lambda = 0$ " and baseline "RTN+EC" in section 5) because quantizing without consideration of the compression mechanism produces poorly compressible quantized weights.

Recent literature includes more advanced network quantization schemes that go beyond a fixed bitwidth per weight, either by combining quantization with pruning [21], or by reserving extra bits for a small percentage of highly salient or outlier weights [11, 13]. However, both the tuning of these extra steps and the binary representation of the required metadata (i.e., how the matrix indices of the pruned or salient weights can be compactly stored in a file) are usually ad-hoc and suboptimal as explicit minimization of the resulting bit rate is infeasible. By contrast, an information-theoretically grounded approach, as proposed in this paper, subsumes both pruning and saliency-dependent quantization as limiting cases, and allows us to interpolate between them by explicitly minimizing a trade-off between the introduced error and the resulting overall bit rate after compression. The next subsection introduces the relevant concepts from information theory and compression that we use in our work.

#### 3.2 Lossy Compression and Information Theory

We consider, for simplicity, the problem of compressing the weight matrix  $\boldsymbol{W} \in \mathcal{W}$  of a single layer of a neural network, where  $\mathcal{W} = \mathbb{R}^{n \times m}$  for some input and output dimensions m and n, respectively. The goal of lossy compression is to find an encoder e that maps  $\boldsymbol{W}$  to a short bit string  $e(\boldsymbol{W}) \in \{0,1\}^* := \bigcup_{\ell=0}^{\infty} \{0,1\}^{\ell}$ , and a corresponding decoder d that maps this bit string to a reconstruction  $\hat{\boldsymbol{W}} := d(e(\boldsymbol{W})) \in \hat{\mathcal{W}}$ , such that  $\hat{\boldsymbol{W}}$  resembles  $\boldsymbol{W}$  with only a small error.<sup>1</sup> Here,  $\hat{\mathcal{W}} \subset \mathcal{W}$  is a typically discrete reconstruction space (discussed in section 3.3 below). To make this problem well-defined, one has to specify a distortion metric  $D: \mathcal{W} \times \hat{\mathcal{W}} \to \mathbb{R}_{\geq 0}$  that quantifies the reconstruction error, and a probabilistic model  $P_{\boldsymbol{W}}$  of the data source, which models prior assumptions that the decoder is allowed to make about the weights before looking at the compressed bit string (for example, the prior assumption that  $\boldsymbol{W}$  is approximately sparse could be expressed by a  $P_{\boldsymbol{W}}$  that models the weights as i.i.d. with a sharp peak around zero). Introducing a Lagrange parameter  $\lambda \geq 0$  that trades off between bit rate and distortion, the optimal encoder/decoder pair  $(e_*, d_*)$  minimizes the following *rate-distortion objective* (where  $|\cdot|$  is the length of the bit string):

$$\operatorname{RD}(\lambda) := \min_{e,d} \mathbb{E}_{\boldsymbol{W} \sim P_{\boldsymbol{W}}} \left[ D(\boldsymbol{W}, d(e(\boldsymbol{W}))) + \lambda |e(\boldsymbol{W})| \right].$$
(1)

Here, the objective takes an expectation over  $P_W$  even though we are ultimately only interested in compressing a specific weight matrix W. This ensures that the resulting decoder  $d_*$  that minimizes Equation 1 only uses the prior assumptions expressed by  $P_W$ , preventing it from simply storing W.

Information content and entropy coding. The minimization problem in Equation 1 is intractable in practice, but we can simplify it by splitting it into a *quantization* step and an *entropy coding* step, where the latter can be solved efficiently [40]. Observe that an optimal decoder  $d_*: \{0, 1\}^* \to \hat{W}$ that solves the minimization problem in Equation 1 is invertible since an encoder/decoder pair that reserves two different bit strings for the same reconstruction  $\hat{W}$  could reduce its expected bit rate by reassigning one of the two bit strings to some  $\hat{W}' \neq \hat{W}$  that is currently assigned to a longer bit string. Thus, with the substitution q(W) := d(e(W)), Equation 1 simplifies to

$$\operatorname{RD}(\lambda) = \min_{q,d} \mathbb{E}_{\boldsymbol{W} \sim P_{\boldsymbol{W}}} \left[ D(\boldsymbol{W}, q(\boldsymbol{W})) + \lambda | d^{-1}(q(\boldsymbol{W})) | \right]$$
(2)

where  $d^{-1}(q(\mathbf{W})) = e(\mathbf{W})$  splits the encoder into a *quantizer*  $q: \mathcal{W} \to \hat{\mathcal{W}}$  and an *entropy coder*  $d^{-1}: \hat{\mathcal{W}} \to \{0, 1\}^*$ , which now no longer appers in the distortion term. Here, the literature on lossless compression provides two important insights. First, the source coding theorem [50, 40] states that the bit rates of an *optimal* entropy coder  $d_*^{-1}$  that minimizes  $\mathbb{E}_{P_{\mathbf{W}}}[|d^{-1}(q(\mathbf{W}))|]$  in Equation 2 for a given quantizer q are given (up to at most 1 bit) by the *information content* of the quantized weights,

$$|d_*^{-1}(\hat{\boldsymbol{W}})| = -\log_2 P_{\hat{\boldsymbol{W}}}(\hat{\boldsymbol{W}}) + \epsilon \qquad \forall \, \hat{\boldsymbol{W}} \in \hat{\mathcal{W}}$$
(3)

where  $\epsilon < 1$  is negligible for any realistically large total bit rate, and the *entropy model*  $P_{\hat{W}}$  is the push-forward of  $P_{W}$  along q (in practice, our proposed method directly models  $P_{\hat{W}}$  instead of  $P_{W}$ ).

The second important insight from the lossless compression literature is that the theoretically optimal bit rate in Equation 3 can very nearly be achieved in practice by computationally efficient entropy coding algorithms such as arithmetic coding [48, 45], range coding [41], or ANS [16, 4]. Thus, the minimization over d in Equation 2 is solved, and the remaining task is to find a quantizer q that

<sup>&</sup>lt;sup>1</sup>More general formulations admit for stochastic en-/decoders. But without an additional constraint such as realism, there always exists a pair of deterministic en-/decoders among the minimizers of Equation 1.

minimizes  $\mathbb{E}_{P_{\boldsymbol{W}}}[D(\boldsymbol{W}, q(\boldsymbol{W})) - \lambda \log_2 P_{\hat{\boldsymbol{W}}}(q(\boldsymbol{W}))]$ . Different to Equations 1 and 2, this problem factorizes over  $\boldsymbol{W} \in \mathcal{W}$ , and so we only need to consider it for the specific weight matrix  $\boldsymbol{W}$  that we actually want to compress. Substituting  $\hat{\boldsymbol{W}} = q(\boldsymbol{W})$ , we arrive at the rate/distortion objective

$$\hat{\boldsymbol{W}}_{*} = \underset{\hat{\boldsymbol{W}}}{\operatorname{arg\,min}} \left[ D(\boldsymbol{W}, \hat{\boldsymbol{W}}) + \lambda R(\hat{\boldsymbol{W}}) \right] \quad \text{with the rate } R(\hat{\boldsymbol{W}}) = -\log_2 P_{\hat{\boldsymbol{W}}}(\hat{\boldsymbol{W}}). \quad (4)$$

**Fractional and sub-1-bit (amortized) bit rates.** Equation 3 provides an analytic expression for the bit rate of an optimal entropy coder. It is hard to overstate the importance of this result as it allows us to attribute how much each matrix element  $\hat{W}_{ij}$  contributes to the total bit rate, even though an optimal entropy coder typically "packs" multiple matrix elements together. For an autoregressive entropy model  $P_{\hat{W}}(\hat{W}) = \prod_{i,j} P_{\hat{W}}(\hat{W}_{ij} | \hat{W}_{<(ij)})$  (where the notation "<(ij)" assumes that some ordering is defined), the total bit rate for encoding the matrix  $\hat{W}$  splits into a sum,  $|d_*^{-1}(\hat{W})| = -\sum_{i,j} \log_2 P_{\hat{W}}(\hat{W}_{ij} | \hat{W}_{<(ij)}) + \epsilon$ , where  $\epsilon < 1$  appears only once outside the sum and can thus be neglected. We can therefore interpret each term in this sum as the (amortized) contribution of an individual matrix element to the total bit rate, and minimizing these individual contributions minimizes the total bit rate  $|d_*^{-1}(\hat{W})|$ . Importantly, these individual amortized bit rates are meaningful even though they are generally non-integer values and are often even below 1 bit in our experiments. Even though such fractional bit rates could not be measured individually by explicitly constructing an optimal entropy coder and encoding a single quantized bit rates by 0.3 bit each would reduce the total bit rate by 30 bit. Being able to quantify bit rates with fractional resolution via Equation 3 is crucial for obtaining good compression performance in the low-bit-rate regime.

### 3.3 Quantization

While constructing a near-optimal entropy coder  $d^{-1}$  in Equation 2 for a given entropy model is a solved problem, efficiently finding an optimal quantizer q is still unsolved. Quantization maps the uncompressed weight matrix  $\mathbf{W} \in \mathbb{R}^{n \times m}$  to  $\hat{\mathbf{W}} = q(\mathbf{W})$  in a discrete reconstruction space  $\hat{\mathcal{W}}$ .

**Quantization grid.** Restricting  $\hat{W}$  to a discrete (i.e., finite or countably infinite) set is unavoidable in (deterministic) compression because the decoder d maps to  $\hat{W}$  from the space of bit strings  $\{0, 1\}^*$ , which is countable. To make our compression method compatible with inference acceleration methods that use integer arithmetic [43], our experiments use  $\hat{W} = G^{n \times m}$ , with a symmetric, uniform grid

$$G = \left\{ i \cdot \|\mathbf{W}\|_{\infty} / ((k-1)/2) \right\}_{i=-(k-1)/2}^{(k-1)/2} \quad \text{for some odd grid size } k \text{ (so that } 0 \in G).$$
(5)

Here, the grid size k = |G| controls how faithful quantization can be in the best case. In the literature [20, 14, 54],  $|G| = 2^r$  is instead often restricted to be a power of two, in which case r would be the number of bits that each quantized matrix element would occupy if its index into G was stored in uncompressed form. This restriction to powers of two is not necessary in our compression method as we use entropy coding to reduce the storage cost of each weight to its actual information content.

More sophisticated designs [12, 57] place more grid points in high density regions of the data distribution, but this usually requires one to explicitly dequantize all quantized matrices before performing operations on them, preventing the use of accelerated inference operations.

Quantization methods. The quantization problem in Equation 4 is a high-dimensional discrete optimization problem, which is infeasible to solve exactly except for very simple distortion metrics D and entropy models  $P_{\hat{W}}$ . Various existing methods can be understood as approximations to this problem. The crudest approximation is *round to nearest* (RTN), i.e., independently setting  $\hat{W}_{ij} = \arg \min_{g \in G} (W_{ij} - g)^2$  for each i, j. Thus, RTN solves Equation 4 for  $\lambda = 0$  and a distortion metric that factorizes over all elements of  $\hat{W}$ . NNCodec [6] extends the approach to  $\lambda > 0$  with the autoregressive entropy model DeepCABAC [55], constructing a greedy approximation to the optimal quantizer q, i.e., when quantizing each matrix element  $W_{ij}$ , NNCodec neglects the effect that the choice of  $\hat{W}_{ij}$  has on the bit rates of subsequent matrix elements by changing the internal state of the autoregressive entropy model. Finally, OPTQ [20] implicitly considers  $\lambda = 0$  again, but uses

as distortion function the non-factorized layerwise loss from [42],  $D(\mathbf{W}, \hat{\mathbf{W}}) := \|\mathbf{W}\mathbf{X} - \hat{\mathbf{W}}\mathbf{X}\|_2^2$ , which considers the euclidean distance of layer outputs rather than of the weights themselves. Here,  $\mathbf{X}$  is the layer input when evaluating the model on a so-called calibration set. OPTQ approximates the minimum of  $D(\mathbf{W}, \hat{\mathbf{W}})$  over  $\hat{\mathbf{W}}$  by iterating over the matrix elements in a fixed order and greedily rounding each element  $W_{ij}$  to the nearest neighbor in G, but it then takes some aspects of the global structure of D into account by updating the remaining (so far unquantized) matrix elements of  $\mathbf{W}$  to compensate, as well as possible, for the error introduced by quantizing  $W_{ij}$ .

Our proposed method CERWU (Compression with Entropy-Regularized Weight Updates), presented in the next section, considers the full rate/distortion objective in Equation 4 with  $\lambda > 0$  and a non-factorized distortion metric D, and it takes the rate term into account both when quantizing each individual weight  $W_{ij}$ , as well as (in an approximate way) during weight updates.

## 4 Method

Our proposed CERWU method starts from the rate/distortion-constrained quantization problem in Equation 4, where we use as the distortion D the layer-wise loss popularized by [42],

$$\hat{\boldsymbol{W}}_* = \operatorname*{arg\,min}_{\hat{\boldsymbol{W}}} L_{\lambda}(\hat{\boldsymbol{W}}) \quad \text{with} \quad L_{\lambda}(\hat{\boldsymbol{W}}) = \|\boldsymbol{W}\boldsymbol{X} - \hat{\boldsymbol{W}}\boldsymbol{X}\|_2^2 + \lambda R(\hat{\boldsymbol{W}}), \quad (6)$$

where  $\boldsymbol{X} \in \mathbb{R}^{m \times p}$  is the layer input when evaluating the model on a calibration set of size p, and the rate  $R(\hat{\boldsymbol{W}}) = -\log_2 P_{\hat{\boldsymbol{W}}}(\hat{\boldsymbol{W}})$  assumes some entropy model  $P_{\hat{\boldsymbol{W}}}$ . The rate term makes  $L_{\lambda}(\hat{\boldsymbol{W}})$ non-quadratic, which complicates its minimization. We therefore split  $L_{\lambda}(\hat{\boldsymbol{W}}) = L'_{\lambda}(\hat{\boldsymbol{W}}) + L''_{\lambda}(\hat{\boldsymbol{W}})$ into a quadratic part  $L'_{\lambda}(\hat{\boldsymbol{W}})$  that approximates  $P_{\hat{\boldsymbol{W}}}(\hat{\boldsymbol{W}})$  with (a discretization of) a Gaussian fit  $\prod_{ij} \mathcal{N}(0, \operatorname{Var}(\{W_{ij}\}_{ij}))$  to the unquantized weights  $\{W_{ij}\}_{ij}$ , and a remainder  $L''_{\lambda}(\hat{\boldsymbol{W}})$ ,

$$L'_{\lambda}(\hat{\boldsymbol{W}}) = \|\boldsymbol{W}\boldsymbol{X} - \hat{\boldsymbol{W}}\boldsymbol{X}\|_{2}^{2} + \frac{\lambda\gamma}{2}\|\hat{\boldsymbol{W}}\|_{2}^{2} \quad \text{and} \quad L''_{\lambda}(\hat{\boldsymbol{W}}) = \lambda R(\hat{\boldsymbol{W}}) - \frac{\lambda\gamma}{2}\|\hat{\boldsymbol{W}}\|_{2}^{2}, \quad (7)$$

with  $\gamma = 1/(\ln(2) \operatorname{Var}(\{W_{ij}\}_{ij}))$ . We can now simplify  $L'_{\lambda}(\hat{\boldsymbol{W}})$  by completing the square,

$$L'_{\lambda}(\hat{\boldsymbol{W}}) = \frac{1}{2} \operatorname{Tr} \left[ (\boldsymbol{W}' - \hat{\boldsymbol{W}}) \boldsymbol{H}' (\boldsymbol{W}' - \hat{\boldsymbol{W}})^T \right] + \text{const.},$$
(8)

where the constant is independent of  $\hat{W}$ . Equation 8 can be verified by multiplying out its r.h.s. (see Appendix C.1) and comparing the result to  $L'(\hat{W})$  in Equation 7, using the entropy-regularized layerwise Hessian H', the unregularized layer-wise Hessian H, and the regularized weight matrix W',

$$H' = H + \lambda \gamma I, \qquad H = 2XX^T, \qquad \text{and} \qquad W' = WH(H')^{-1}.$$
 (9)

To obtain a quantized matrix  $\hat{\boldsymbol{W}} \in G^{n \times m}$  (the uniform grid G is defined in Equation 5) that approximately minimizes  $L_{\lambda}(\hat{\boldsymbol{W}}) = L'_{\lambda}(\hat{\boldsymbol{W}}) + L''_{\lambda}(\hat{\boldsymbol{W}})$ , we now iteratively apply the optimal brain surgeon (OBS) algorithm [25] to the quadratic part  $L'_{\lambda}(\hat{\boldsymbol{W}})$ , interleaving it with a rate-constrained quantization step that takes the non-quadratic part  $L''_{\lambda}(\hat{\boldsymbol{W}})$  into account as well.

Entropy-regularized weight updates. Algorithm 1 summarizes our proposed CERWU algorithm. Following [20], we iterate over the rows *i* and columns *j* of the weight matrix without optimizing over the iteration order. For each (i, j), we first obtain  $\hat{W}_{ij} \in G$  by quantizing  $W'_{ij}$  as described below, and we then update so-far unquantized matrix elements of W' to optimally compensate for the error introduced by quantizing  $W'_{ij}$ . Since a closed-form solution for this update is only available for a quadratic loss function, we consider only the quadratic approximation  $L'_{\lambda}(\hat{W})$  in Equation 8 for the weight update, assuming that the non-quadratic remainder  $L''_{\lambda}(\hat{W})$  is small by construction. Adapting the results in [25] (see also derivation in Appendix C) to our quadratic loss  $L'_{\lambda}(\hat{W})$ , only weights  $W'_{i,>j}$  in the same row *i* need to be updated, and the optimal update  $\Delta_{W'_{i,>j}}$  and the incurred increase  $\Delta_{L'}$  of the quadratic part of the loss due to quantizing  $W'_{ij}$  and updating  $W'_{i,>j}$  are

$$\Delta_{\mathbf{W}'_{i,>j}} = -\frac{W'_{ij} - \hat{W}_{ij}}{\left[(\mathbf{H}'_{\geq j,\geq j})^{-1}\right]_{jj}} \left[(\mathbf{H}'_{\geq j,\geq j})^{-1}\right]_{j,>j} \quad \text{and} \quad \Delta_{L'} = \frac{1}{2} \frac{(W'_{ij} - \hat{W}_{ij})^2}{\left[(\mathbf{H}'_{\geq j,\geq j})^{-1}\right]_{jj}}.$$
 (10)

Note that the weight update  $\Delta_{W'_{i,>j}}$  (as well as the initialization of W' in Equation 9) depends on the *entropy-regularized* Hessian H', which contains the term  $\lambda \gamma I \propto \lambda / \operatorname{Var}(\{W_{ij}\}_{ij})$  that becomes large for a strong rate constraint  $\lambda$  and for small unquantized weights. This regularization prevents the weights from accumulating large values over the course of repeated weight updates, which can hurt compressibility (see ablation "CERWU- $\gamma = 0$ " in section 5).

Following [20] (see also derivation in section C.5), we pre-compute all required inverse Hessian entries in Equation 10 using the Cholesky decomposition  $C' := \text{Cholesky}((H')^{-1})^T$  to improve runtime and numerical stability, resulting in the equivalent equations

$$\Delta_{\mathbf{W}'_{i,>j}} = -\frac{W'_{ij} - \hat{W}_{ij}}{C'_{jj}} \mathbf{C}'_{j,>j} \quad \text{and} \quad \Delta_{L'} = \frac{1}{2} \frac{(W'_{ij} - \hat{W}_{ij})^2}{C'_{jj}^2}.$$
 (11)

**Quantization.** For each (i, j), before updating future weights  $W'_{i,>j}$ , we quantize the current weight  $W'_{ij}$  to  $\hat{W}_{ij} \in G$ . Since the grid size |G| is fixed, we can afford to explicitly search over all  $g \in G$  and thus consider, in addition to  $\Delta_{L'}$  from Equation 11, also changes to the non-quadratic part  $L''(\hat{W})$  of the layer-wise loss (Equation 7). However, only the contribution of the current matrix entry  $\hat{W}_{ij}$  to  $L''(\hat{W})$  can be taken into account efficiently, and we have to neglect second-order effects to  $L''(\hat{W})$  due to resulting updates for  $W'_{i,>j}$ . Assuming an autoregressive entropy model  $P_{\hat{W}}(\hat{W}) = \prod_{i,j} P_{\hat{W}}(\hat{W}_{ij} | \hat{W}_{<(ij)})$ , the contribution of  $\hat{W}_{ij}$  to the rate  $R(\hat{W}) = -\log_2 P_{\hat{W}}(\hat{W})$  is  $-\log_2 P_{\hat{W}}(\hat{W}_{ij} | \hat{W}_{<(ij)})$ , and thus our quantization q sets  $\hat{W}_{ij} \leftarrow q(W'_{ij}, \lambda, \gamma, P_{\hat{W}}(\cdot | \hat{W}_{<(ij)}))$  with

$$q(W'_{ij}, \lambda, \gamma, P) := \operatorname*{arg\,min}_{g \in G} \left[ \frac{1}{2} \frac{(W'_{ij} - g)^2}{C'_{jj}^2} - \lambda \log_2 P(g) - \frac{\lambda \gamma}{2} g^2 \right]$$
(12)

**Scan order.** When using an autoregressive entropy model, the order in which the weights are fed to the model starts to matter. There exist two different possibilities, *row-major* (traversing along rows first) and *column-major* (columns first). Although we did not observe a large performance difference between these two, we include both in our experiments and select the best version for each method.

**Complexity analysis.** Running the algorithm for a full network first requires a single forward pass over a moderately sized calibration set (e.g., we used 40'000 data points for ImageNet, around 3% of the full training data set). The calculated activations can be cached for each layer, and future runs to quantize the network with different compression strengths  $\lambda$  do not need to repeat this forward pass.

As for the quantization algorithm itself, assume a grid of size k. Each layer of

Algorithm 1 Compression with Entropy-Regularized Weight Updates (CERWU) (row-major variant).

- **Input:** uncompressed weight matrix  $W \in \mathbb{R}^{n \times m}$ ; grid size k (see Equation 5); Hessian  $H = 2XX^T \in \mathbb{R}^{m \times m}$  (see Equation 9); rate/distortion trade-off parameter  $\lambda > 0$ ; autoreg. entropy model P (e.g., DeepCABAC [55]).
- **Output:** quantized weights  $\hat{W} \in G^{n \times m}$  (see Equation 5) that can be well compressed with the entropy model *P*.
- 1: Set  $\gamma \leftarrow 1/(\ln(2) \operatorname{Var}(\{W_{ij}\}_{ij})) \triangleright$  See Equation 7. 2: Set  $H' \leftarrow H + \lambda \gamma I$
- 3: Initialize  $W' \leftarrow WH(H')^{-1} \implies See Equation 9.$
- 4: Set  $C' \leftarrow \text{Cholesky}((H')^{-1})^T \triangleright (upper triangular)$
- 5: Initialize  $P \leftarrow initEntropyModel()$
- 6: for row i = 1 to n do
- 7: **for** column j = 1 to m **do** 8: Set  $\hat{W}_{ij} \leftarrow q(W'_{ij}, \lambda, \gamma, P) \triangleright See Equation 12.$

Update 
$$W'_{i,>j} \leftarrow W'_{i,>j} - \frac{W_{ij} - W_{ij}}{C'_{jj}}C'_{j,>j}$$
  
  $\triangleright$  See Equation 11.

- 10: *P*.autoregressiveUpdate( $\hat{W}_{ij}$ )
- 11: end for12: end for

9:

size  $n \times m$  requires  $O(m^3)$  operations to calculate the Cholesky decomposition of the  $m \times m$  sized inverse regularized Hessian  $(\mathbf{H}')^{-1}$ . The quantization procedure requires O(nm) quantization steps, each of which require an O(k) grid search followed by an O(m) row update. Therefore, the full complexity is  $O(m^3 + nm^2 + nmk)$ . Usually, it holds that  $k \ll m$  for most layers, however, the grid search requires O(k) calls to the entropy model. The cost for a single evaluation of the entropy model usually does not depend on m, n, k, but can still be expensive, depending on the chosen model. We show empirical runtimes in section 5.2.

# **5** Experiments

**Experiment setup.** We test our proposed quantization method on multiple networks from the computer vision community. We include ResNet-{18, 52, 101, 152} [26], VGG16 [52] and MobileNetv3-{small, large} [29] trained on ImageNet [10], as well as ResNet-{18, 34, 52} trained on CIFAR10 [34]. For our entropy model, we use DeepCABAC [55] due to its high speed and its focus on modeling typical neural network weight statistics. For all methods, we perform a sweep over the scan-order, various compression strength parameters and grid sizes to ensure a good coverage of the rate-distortion curve. Shown bit rates are the measured bit rates after actual entropy coding plus a small overhead for storing the grid spacings for each layer. Refer to Appendix A for more detailed information.

**Pareto front.** To visualize the results from our search space, we calculate a Pareto front in the rate-distortion space in the following manner: for a given network and compression method, we iterate over all data points (rate, accuracy) obtained from our sweep. For each data point x, if there exists a point y that has both a lower rate as well as higher or equal accuracy, we discard x.

Ablations and baselines. Aside from our main proposed compression algorithm detailed in Algorithm 1, we also explore two variations: a version with unregularized weight updates, which amounts to artificially setting  $\gamma = 0$ , and a version that does not account for the rate during quantization, which amounts to setting  $\lambda = 0$  (this also renders the method independent of  $\gamma$ ). The case of  $\lambda = 0$  can also be interpreted as performing OPTQ [20] followed by entropy coding with the chosen entropy model. These two methods are labeled as CERWU- $\gamma = 0$  and CERWU- $\lambda = 0$  respectively. We also compare our method against the state-of-the-art compression scheme NNCodec [6], an implementation of the Neural Network Compression Standard ISO/IEC 15938-17 also based on the DeepCABAC entropy model. As a further baseline, we include a simple quantization scheme of nearest-neighbour quantization followed with entropy coding via DeepCABAC, which we label as RTN+EC.

### 5.1 Compression Performance

Figure 1 presents a comparison of the three variants of our proposed compression method and the baselines on our analyzed networks for ImageNet. Figure 2 shows the same for networks trained on CIFAR10. Additionally, Figure 3 shows bar plots of the bit rate achieved by each method for the strongest compression setting that still retains 99% and 95%, respectively, of the original performance.

We observe that CERWU and CERWU- $\gamma = 0$  consistently outperform all other tested methods, achieving a more favorable rate-distortion performance over most of the rate-distortion curve. While in the line plots it can be hard to distinguish between CERWU and CERWU- $\gamma = 0$ , the bar plots can help distinguish between them. There, we see that CERWU maintains a slight but significant edge (1-5% stronger compression performance) for most networks for both evaluated fixed performance levels over CERWU- $\gamma = 0$ . This demonstrates that taking estimations of future rates into account when performing the weight updates helps to create more strongly compressible representations.

Compared to the baselines, CERWU and CERWU- $\gamma = 0$  often achieve much stronger compression rates when compressing the network to almost-original performance (which we expect to be the regime of interest for most practical applications), producing a compressed representation that is 20-40% more strongly compressed than the current compression standard NNCodec, see Figure 3.

# 5.2 Run times

**Encoding times.** We timed our proposed compression method CERWU on our hardware (described in Appendix A) for ImageNet-trained ResNets of varios sizes (the size of the training set does not influence encoding times). Initial runs (Figure 4 left) include the time for the forward passes to calculate the Hessian. Subsequent runs (Figure 4 right) are considerably faster as they can reuse the Hessian H for different values of  $\lambda$  or k. Especially in the regime of small grid sizes  $k (\leq 31, \text{ which}$  are usually sufficient to produce high-accuracy compressed representations), the run time is heavily dominated by the calculation of the Hessian, and runs after the first one become quite cheap. The baselines NNCodec and RTN+EC only require between 1-10 seconds to quantize each network. This is much faster than our method for large grids or for the first run. While our achieved encoding times are still in the manageable realm, further work to speed up the grid search might be helpful (such as introducing early stopping), as well as a thorough optimization of our implementation.



Figure 1: Performance of our compression methods on various networks trained and evaluated on ImageNet. For better visibility, a Pareto front over the parameters was calculated for each curve. The inset at the bottom right of each plot shows a zoomed-in version of the area marked in the red bounding box. The inset ranges over  $(0.95 \cdot acc_{orig.}, 1.0125 \cdot acc_{orig.})$  on the y-axis and encompasses a 1.5-bits-per-weight range in the x-axis. Our proposed methods and ablations are marked with solid lines, baselines are marked in dashed lines, and the original performance of the (uncompressed) network is marked with a horizontal, gray dashed line. The plot titles include the number of quantizable parameters each network has (multiply by 32 bit to get the uncompressed storage size on disk).



Figure 2: Performance of compression methods for CIFAR10-trained networks, analogous to Figure 1.

**Decoding times.** The *decoding time* is the more important metric, as often compression is performed once on a device with larger computational resources, while decompression is performed many times on weaker devices. Here, by construction of our method, we achieve the same decompression speed as NNCodec or RTN+EC, since we can decompress our network using the same decoder in form of the DeepCABAC entropy model (the decompression algorithm is oblivious to the distortion function and quantization scheme used during compression). The decompression speed is between 0.2 and 1.2 seconds for all networks and all tested methods on our hardware (see Appendix A) despite using a single-threaded decoder implementation.



Figure 3: Minimum bits per weight achieved at 99% (left) and 95% (right) of the original test accuracy for different methods. Lower is better. Both CERWU (blue) and CERWU- $\gamma = 0$  (orange) outperform all other methods, with CERWU achieving a slight edge over CERWU- $\gamma = 0$  for most networks.



Figure 4: Run times for compressing ResNets of differing sizes. Left: run times for the first run; Right: run times for subsequent runs for different values of  $\lambda$  or k (which can reuse the Hessian).

# 6 Conclusion

We showcased a post-training compression method that achieves state-of-the-art storage sizes for computer vision architectures suitable for resource constrained devices. We showed how adding a quadratic rate estimation to the layerwise loss provides locally exact solutions to the resulting rate/distortion constrained weight update problem, extending the OBS framework on which many contemporary compression works build [18, 20, 19]. Our experiments demonstrate a 20-40% reduction in file size at no additional accuracy drop compared to the state-of-the-art method NNCodec.

Moreover, the flexibility of our method opens up interesting avenues for future research directions and applications. For example, our method can be used with different entropy models and quantization grids, and the resulting compression performance can be evaluated. Additionally, given a suitable entropy model, our method can also produce compressible representations that could be decoded *on-the-fly* on a GPU. This might serve to reduce the required communication bandwidth between RAM and GPU during inference, resulting in possible energy savings and improved latency.

**Limitations.** While our work shows strong compression performance on (convolutional) computer vision architectures, we do not include an analysis for transformer-based models. We argue that resource-constrained devices often resort to simpler architectures such as those that we surveyed due to their weaker computational power. In particular, we do not discuss (large) language models, where some quick experiments (see Appendix B) indicate worse performance of both our method and NNCodec, but where we argue that there already exists a large body of work that specifically targets these models [2, 17, 53, 15, 1], often taking specific architecture-dependent behavior into account (e.g., outliers in the channel activations [39, 31]).

## Acknowledgements

We thank Tim Z. Xiao for pointing us to relevant literature, and Julia Dietl for insightful discussions and an early exploration of model compression methods.

This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC number 2064/1 – Project number 390727645. This work was supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A. RB acknowledges funding by the German Research Foundation (DFG) for project 448588364 of the Emmy Noether Program.

### References

- Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs, October 2024.
- [2] Mart Van Baalen, Andrey Kuzmin, Markus Nagel, Peter Couperus, Artem Bolshakov, Cedric Bastoul, Eric Mahurin, Tijmen Blankevoort, and Paul Whatmough. GPTVQ: The blessing of dimensionality for LLM quantization. In Workshop on Efficient Systems for Foundation Models II, International Conference on Machine Learning (ICML), 2024.
- [3] Junjie Bai, Fang Lu, Ke Zhang, et al. ONNX: Open neural network exchange, 2019.
- [4] Robert Bamler. Understanding entropy coding with asymmetric numeral systems (ans): a statistician's perspective. *arXiv preprint arXiv:2201.01741*, 2022.
- [5] Chaim Baskin, Brian Chmiel, Evgenii Zheltonozhskii, Ron Banner, Alex M. Bronstein, and Avi Mendelson. CAT: Compression-Aware Training for bandwidth reduction. *Journal of Machine Learning Research*, 22(269):1–20, 2021.
- [6] Daniel Becking, Paul Haase, Heiner Kirchhoffer, Karsten Müller, Wojciech Samek, and Detlev Marpe. NNCodec: An Open Source Software Implementation of the Neural Network Coding ISO/IEC Standard. In Workshop Neural Compression: From Information Theory to Applications, International Conference on Machine Learning (ICML), July 2023.
- [7] Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. EfficientQAT: Efficient Quantization-Aware Training for Large Language Models, October 2024.
- [8] Brian Chmiel, Chaim Baskin, Ron Banner, Evgenii Zheltonozhskii, Yevgeny Yermolin, Alex Karbachevsky, Alex M. Bronstein, and Avi Mendelson. Feature Map Transform Coding for Energy-Efficient CNN Inference, September 2019.
- [9] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Universal Deep Neural Network Compression. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):715–726, May 2020.
- [10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A largescale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.
- [11] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. GPT3.int8(): 8-bit Matrix Multiplication for Transformers at Scale. *Neural Information Processing Systems (NeurIPS)*, January 2022.
- [12] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs, May 2023.
- [13] Tim Dettmers, Ruslan A. Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A Sparse-Quantized Representation for Near-Lossless LLM Weight Compression. In *The Twelfth International Conference on Learning Representations*, October 2023.

- [14] Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: K-bit Inference Scaling Laws. In *Proceedings of the 40th International Conference on Machine Learning*, pages 7750–7774. PMLR, July 2023.
- [15] Peijie Dong, Lujun Li, Dayou Du, Yuhan Chen, Zhenheng Tang, Qiang Wang, Wei Xue, Wenhan Luo, Qifeng Liu, Yike Guo, and Xiaowen Chu. STBLLM: Breaking the 1-Bit Barrier with Structured Binary LLMs, August 2024.
- [16] Jarek Duda, Khalid Tahboub, Neeraj J Gadgil, and Edward J Delp. The use of asymmetric numeral systems as an accurate replacement for huffman coding. In 2015 Picture Coding Symposium (PCS), pages 65–69. IEEE, 2015.
- [17] Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme Compression of Large Language Models via Additive Quantization, September 2024.
- [18] Elias Frantar and Dan Alistarh. Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning. In Advances in Neural Information Processing Systems (NeurIPS), volume 35, pages 4475–4488, December 2022.
- [19] Elias Frantar and Dan Alistarh. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot, March 2023.
- [20] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. OPTQ: Accurate quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*, 2023.
- [21] Elias Frantar, Utku Evci, Wonpyo Park, Neil Houlsby, and Dan Alistarh. Compression Scaling Laws:Unifying Sparsity and Quantization, February 2025.
- [22] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge Distillation of Large Language Models. In *The Twelfth International Conference on Learning Representations*, October 2023.
- [23] Song Han, Huizi Mao, and William J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, February 2016.
- [24] Yongchang Hao, Yanshuai Cao, and Lili Mou. NeuZip: Memory-Efficient Training and Inference with Dynamic Compression of Neural Networks, October 2024.
- [25] B. Hassibi, D.G. Stork, and G.J. Wolff. Optimal Brain Surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299 vol.1, March 1993.
- [26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, June 2016.
- [27] Fred Hohman, Mary Beth Kery, Donghao Ren, and Dominik Moritz. Model Compression in Practice: Lessons Learned from Practitioners Creating On-device Machine Learning Experiences. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–18, May 2024.
- [28] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization, July 2024.
- [29] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324, 2019.
- [30] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate Post Training Quantization With Small Calibration Sets. In *Proceedings of the 38th International Conference on Machine Learning*, pages 4466–4475. PMLR, July 2021.

- [31] Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-Sparse Quantization, February 2024.
- [32] Heiner Kirchhoffer, Paul Haase, Wojciech Samek, Karsten Müller, Hamed Rezazadegan-Tavakoli, Francesco Cricri, Emre B. Aksu, Miska M. Hannuksela, Wei Jiang, Wei Wang, Shan Liu, Swayambhoo Jain, Shahab Hamidi-Rad, Fabien Racapé, and Werner Bailer. Overview of the Neural Network Compression and Representation (NNR) Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(5):3203–3216, May 2022.
- [33] Jong Hwan Ko, Duckhwan Kim, Taesik Na, Jaeha Kung, and Saibal Mukhopadhyay. Adaptive weight compression for memory-efficient neural networks. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 199–204, March 2017.
- [34] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).
- [35] Mounssif Krouka, Anis Elgabli, Chaouki Ben Issaid, and Mehdi Bennis. Energy-Efficient Model Compression and Splitting for Collaborative Inference Over Time-Varying Channels. In 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pages 1173–1178, September 2021.
- [36] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention, September 2023.
- [37] Bingrui Li, Jianfei Chen, and Jun Zhu. Memory Efficient Optimizers with 4-bit States. Advances in Neural Information Processing Systems, 36:15136–15171, December 2023.
- [38] Shiyu Li, Edward Hanson, Hai Li, and Yiran Chen. PENNI: Pruned Kernel Sharing for Efficient CNN Inference. In *International Conference on Machine Learning (ICML)*, pages 5863–5873. PMLR, November 2020.
- [39] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration, April 2024.
- [40] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms.* Cambridge university press, 2003.
- [41] G Nigel N Martin. Range encoding: an algorithm for removing redundancy from a digitised message. In Proc. Institution of Electronic and Radio Engineers International Conference on Video and Data Recording, page 48, 1979.
- [42] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or Down? Adaptive Rounding for Post-Training Quantization. In *International Conference* on Machine Learning (ICML), pages 7197–7206, November 2020.
- [43] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A White Paper on Neural Network Quantization, June 2021.
- [44] Youngsuk Park, Kailash Budhathoki, Liangfu Chen, Jonas M. Kübler, Jiaji Huang, Matthäus Kleindessner, Jun Huan, Volkan Cevher, Yida Wang, and George Karypis. Inference Optimization of Foundation Models on AI Accelerators. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pages 6605–6615, New York, NY, USA, August 2024. Association for Computing Machinery.
- [45] Richard Clark Pasco. *Source coding algorithms for fast data compression*. PhD thesis, Stanford University CA, 1976.
- [46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, December 2019.

- [47] Haotong Qin, Xudong Ma, Xingyu Zheng, Xiaoyang Li, Yang Zhang, Shouda Liu, Jie Luo, Xianglong Liu, and Michele Magno. Accurate LoRA-Finetuning Quantization of LLMs via Information Retention, May 2024.
- [48] Jorma Rissanen and Glen G Langdon. Arithmetic coding. *IBM Journal of research and development*, 23(2):149–162, 1979.
- [49] Kavya Saravanan and Abbas Z. Kouzani. Advancements in On-Device Deep Neural Networks. Information, 14(8):470, August 2023.
- [50] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, July 1948.
- [51] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Re, Ion Stoica, and Ce Zhang. FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU. In *Proceedings of the 40th International Conference on Machine Learning*, pages 31094–31116. PMLR, July 2023.
- [52] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, 2015.
- [53] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A Simple and Effective Pruning Approach for Large Language Models, May 2024.
- [54] Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even Better LLM Quantization with Hadamard Incoherence and Lattice Codebooks, June 2024.
- [55] Simon Wiedemann, Heiner Kirchoffer, Stefan Matlage, Paul Haase, Arturo Marban, Talmaj Marinc, David Neumann, Tung Nguyen, Ahmed Osman, Detlev Marpe, Heiko Schwarz, Thomas Wiegand, and Wojciech Samek. DeepCABAC: A Universal Compression Algorithm for Deep Neural Networks. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):700–714, May 2020.
- [56] Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Compact and computationally efficient representation of deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(3):772–785, 2020.
- [57] Yibo Yang, Robert Bamler, and Stephan Mandt. Variational bayesian quantization. In International Conference on Machine Learning, pages 10670–10680. PMLR, 2020.
- [58] J. Ziv. On universal quantization. *IEEE Transactions on Information Theory*, 31(3):344–347, May 1985.

# **A** Experimental Details

**Evaluation details.** For the tested networks, we report the accuracy, which is the top-1 accuracy and the average number of bits required to encode one weight. The bits per weight are obtained by dividing the total compressed size by the number of compressed parameters, while the total compressed size consists of the size of the entropy coded weights plus the overhead of one 16-bit scale parameter per tensor. For ResNets, we don't quantize the last layer, as well as the BatchNorm layers, as they can often be fused together with preceding layers. To ensure a fair comparison, these layers are not quantized for all analyzed methods. To calculate the layer-wise loss for convolutional layers, we unfold convolutional layers (see f.e. [42, 30, 18]) For the MobileNet networks to be compressed with our method, we pre-process the grouped convolutions to be represented as  $n_{\text{groupsize}}$  single-filter convolutions, whose results are then concatenated (this does not change the network output). We then compress each convolution separately, taking care to count one scale parameter per filter to the total compression overhead.

**Parameter choices.** To calculate the hessians, we use calibration samples from the training set containing 40'000 images for ImageNet and 64'000 images for CIFAR10. For ImageNet, we use 16'000 samples from the validation set to calculate the top-1 accuracy, and for CIFAR10, we use 10'000 samples from the test set. For the grid, we use a symmetric uniform grid and sweep between a subset of grid sizes {4,6,8,12,16,32,48,64,128,256,512,1024}, depending on the method and network. For the methods that include a  $\lambda$  parameter, we perform a sweep between values of  $10^{-8}$  and  $10^{-1}$  at steps of 0.5 in log-space, with increased granularity of 0.1 in regions of high variability. Additionally, for every method, we once iterate over the weights in row-major and once in column-major order. For NNCodec, we swept over the qp parameter in a (-38, -4) interval.

**Soft- and hardware.** Our algorithm is implemented mainly in C++, while the code associated with model loading and evaluation was implemented using the PyTorch library [46], whose pre-trained models for ImageNet were used for our evaluations. The pre-trained ResNet models for CIFAR10 were obtained from a public repository of user edaltocg on HuggingFace. The quantization part of the algorithm was executed on a 2.6 GHz Intel Xeon Gold 6240, while the evaluation and calculation of the Hessian were done on a single NVIDIA 2080ti graphics card.



# **B** Pythia Evaluations.

Figure 5: Rate-distortion performance of our methods on the small language model Pythia-70M. We swept over grid sizes of {4, 16, 128, 256, 352, 512, 768, 1024, 2048}.

# C Derivation of the Entropy-Regularized Weight Update

This section provides a derivation of Equations 10 and 11 in compact form. We stress that the derivation is not new, but rather combines arguments by Hassibi et al. in [25] and by Frantar et

al. in [20], applying them to the quadratic part  $L'_{\lambda}(\hat{W})$  of our entropy-regularized layerwise loss function in Equation 7. We provide the derivation here for completeness, to clarify what it means precisely when we say that the weight updates "optimally compensate" for introduced rounding errors, to show how analogous arguments as in [25, 20] apply to our modified loss function, and to motivate why the weight updates in our proposed CERWU method take the rate term into account only within a quadratic approximation (as opposed to the quantization in Equation 12, which also takes the non-quadratic part  $L''_{\lambda}(\hat{W})$  into account).

### C.1 Verification of Equation 8

Before we formalize what we mean with "optimal weight updates", we first verify that Equation 8 is indeed equivalent to the definition of  $L'_{\lambda}(\hat{W})$  in Equation 7. Using the squared Frobenius norm  $\|A\|_2^2 := \sum_{i,j} A_{ij}^2 = \text{Tr}[AA^T]$  for any matrix A, and  $H := 2XX^T$ , we multiply out the right-hand side of Equation 7 and find,

r.h.s. of Equation 7 = 
$$\|(\boldsymbol{W} - \hat{\boldsymbol{W}})\boldsymbol{X}\|_{2}^{2} + \frac{\lambda\gamma}{2}\|\hat{\boldsymbol{W}}\|_{2}^{2}$$
  
=  $\frac{1}{2}\operatorname{Tr}\left[(\boldsymbol{W} - \hat{\boldsymbol{W}})\boldsymbol{H}(\boldsymbol{W} - \hat{\boldsymbol{W}})^{T} + \lambda\gamma\hat{\boldsymbol{W}}\hat{\boldsymbol{W}}^{T}\right]$   
=  $\frac{1}{2}\operatorname{Tr}\left[\underbrace{\boldsymbol{W}\boldsymbol{H}\boldsymbol{W}^{T}}_{\text{const.}} - 2\boldsymbol{W}\boldsymbol{H}\hat{\boldsymbol{W}}^{T} + \hat{\boldsymbol{W}}\boldsymbol{H}\hat{\boldsymbol{W}}^{T} + \lambda\gamma\hat{\boldsymbol{W}}\hat{\boldsymbol{W}}^{T}\right]$  (13)

where the "const." term is independent of  $\hat{W}$ . By comparison, inserting  $H' = H + \lambda \gamma I$  and  $W' = WH(H')^{-1}$  from Equation 9 into Equation 8, we find (using that H and H' are symmetric):

r.h.s. of Equation 8 = 
$$\frac{1}{2} \operatorname{Tr} \left[ (\mathbf{W}' - \hat{\mathbf{W}}) \mathbf{H}' (\mathbf{W}' - \hat{\mathbf{W}})^T \right] + \operatorname{const.}$$
  
=  $\frac{1}{2} \operatorname{Tr} \left[ \left( \mathbf{W} \mathbf{H} (\mathbf{H}')^{-1} - \hat{\mathbf{W}} \right) \mathbf{H}' \left( (\mathbf{H}')^{-1} \mathbf{H} \mathbf{W}^T - \hat{\mathbf{W}}^T \right) \right] + \operatorname{const.}$   
=  $\frac{1}{2} \operatorname{Tr} \left[ \underbrace{\mathbf{W} \mathbf{H} (\mathbf{H}')^{-1} \mathbf{H} \mathbf{W}^T}_{\operatorname{const. (independent of \hat{\mathbf{W}})}} - 2\mathbf{W} \mathbf{H} \hat{\mathbf{W}}^T + \hat{\mathbf{W}} \mathbf{H}' \hat{\mathbf{W}}^T \right] + \operatorname{const.}$   
=  $\frac{1}{2} \operatorname{Tr} \left[ -2\mathbf{W} \mathbf{H} \hat{\mathbf{W}}^T + \hat{\mathbf{W}} (\mathbf{H} + \lambda \gamma \mathbf{I}) \hat{\mathbf{W}}^T \right] + \operatorname{const.}$  (14)

Thus, Equations 7 and 8 are indeed equivalent. The advantage of Equation 8 is that its simpler form allows us to immediately see that the Hessian of  $L'(\hat{W})$  is H', and that  $L'(\hat{W})$  would be minimized if we could set  $\hat{W}$  to the entropy regularized weight matrix  $W' = WH(H')^{-1}$  (which, however, is not possible because  $W' \notin G^{n \times m}$  in general).

### C.2 Formalization of the Propositions

For  $W' \in \mathbb{R}^{n \times m}$  and  $\hat{W} \in G^{n \times m}$ , the quadratic loss  $L'_{\lambda}(\hat{W})$  in Equation 8 separates over the rows,

$$L'_{\lambda}(\hat{\boldsymbol{W}}) = \sum_{i=1}^{n} L'_{i,\lambda}(\hat{\boldsymbol{W}}_{i,:}) + \text{const.}, \quad L'_{i,\lambda}(\hat{\boldsymbol{W}}_{i,:}) = \frac{1}{2} (\boldsymbol{W}'_{i,:} - \hat{\boldsymbol{W}}_{i,:}) \boldsymbol{H}' (\boldsymbol{W}'_{i,:} - \hat{\boldsymbol{W}}_{i,:})^{T} \quad (15)$$

where the notation  $A_{i,:}$  denotes a row vector comprised of the *i*-th row of a matrix A, and  $H' \in \mathbb{R}^{m \times m}$  (see Equation 9) is symmetric and positive definite. Due to the separation over rows, minimizing  $L'_{\lambda}(\hat{W})$  over  $\hat{W}$  is equivalent to independently minimizing  $L'_{i,\lambda}(\hat{W}_{i,:})$  for each row  $i \in \{1, \ldots, n\}$ , and we therefore focus the remaining discussion on a given row *i*. Our goal is to show that the weight updates in Equation 10 and Equation 11 are equivalent, and that they optimally compensate subsequent weights in the same row *i* for the error introduced by quantizing  $W'_{ij}$  to  $\hat{W}_{ij}$ .

More formally, consider starting with the unquantized row  $W_{i,:}^{\prime(0)} := W_{i,:}^{\prime} \in \mathbb{R}^{1 \times m}$ , and iterating over its entries  $j \in \{1, \ldots, m\}$  in ascending order. At each iteration j, we construct an updated row vector  $W_{i,:}^{\prime(j)} \in \mathbb{R}^{1 \times m}$  by leaving the first j - 1 entries unchanged (i.e.,  $W_{i,< k}^{\prime(j)} := W_{i,< k}^{\prime(j-1)} = \hat{W}_{i,< k}$ ),

quantizing the *j*-th entry  $W_{ij}^{\prime(j)} := \hat{W}_{ij}$  to some (for the sake of the proof below) arbitrary value  $\hat{W}_{ij}$ , and setting the remaining m - j entries to

$$\boldsymbol{W}_{i,>j}^{\prime(j)} := \operatorname*{arg\,min}_{\tilde{\boldsymbol{W}}_{i,>j} \in \mathbb{R}^{1 \times (m-j)}} L_{i,\lambda}^{\prime}(\tilde{\boldsymbol{W}}_{i,:}) \quad \text{with the constraint} \quad \tilde{\boldsymbol{W}}_{i,\leq j} = \hat{\boldsymbol{W}}_{i,\leq j}, \quad (16)$$

where we formally extend the domain of  $L'_{i,\lambda}$  from  $G^{1\times m}$  to  $\mathbb{R}^{1\times m}$  according to Equation 15.

**Propositions.** For  $j \in \{1, ..., m\}$  and  $W_{i,>j}^{\prime(j)}$  given in Equation 16, we claim that

- (i)  $W_{i,>j}^{\prime(j)} = W_{i,>j}^{\prime(j-1)} + \Delta_{W_{i,>j}}$  where  $\Delta_{W_{i,>j}}$  is given in Equation 10;
- (ii) the quantization and subsequent weight update increase the quadratic loss by

$$L'_{i,\lambda}(\boldsymbol{W}_{i,:}^{\prime(j)}) - L'_{i,\lambda}(\boldsymbol{W}_{i,:}^{\prime(j-1)}) = \Delta_{L'}$$
(17)

where  $\Delta_{L'}$  is also given in Equation 10; and

(iii) Equations 10 and 11 are equivalent expressions for  $\Delta_{W'_{i>i}}$  and  $\Delta_{L'}$ .

### C.3 Proof of Proposition (i)

We prove proposition (i) by induction over  $j \in \{0, ..., m\}$ . For j = 0, we already defined  $W_{i,:}^{\prime(0)} := W_{i,:}^{\prime}$ , which is trivially equivalent to Equation 16 because  $L_{i,\lambda}^{\prime}$  is positive definite with its minimum at  $L_{i,\lambda}^{\prime}(W_{i,:}^{\prime}) = 0$ . Consider now some  $j \in \{1, ..., m\}$  and assume that  $W_{i,<k}^{\prime(j-1)} = \hat{W}_{i,<k}$ , and that Equation 16 holds for j - 1, i.e.,

$$\boldsymbol{W}_{i,\geq j}^{\prime(j-1)} = \operatorname*{arg\,min}_{\tilde{\boldsymbol{W}}_{i,\geq j} \in \mathbb{R}^{1 \times (m-j+1)}} L_{i,\lambda}^{\prime}(\tilde{\boldsymbol{W}}_{i,:}) \qquad \text{with the constraint} \qquad \tilde{\boldsymbol{W}}_{i,< j} = \hat{\boldsymbol{W}}_{i,< j} \qquad (18)$$

(note the " $\geq$ " instead of ">", and "<" instead of " $\leq$ " when comparing Equation 18 to Equation 16). We simplify Equation 18 by splitting  $L'_{i,\lambda}(\tilde{W}_{i,:})$  from Equation 15 into four parts:

$$L_{i,\lambda}'(\tilde{\boldsymbol{W}}_{i,:}) = \frac{1}{2} \begin{pmatrix} \boldsymbol{W}_{i,(19)$$

where we wrote W' explicitly as  $W'^{(0)}$  to stress that  $L'_{i,\lambda}$  always compares to the *original* unquantized row, not to any updated version. By Equation 18, we know that  $W'^{(j-1)}_{i,:}$  is a stationary point of  $L'_{i,\lambda}(\tilde{W}_{i,:})$  with respect to  $\tilde{W}_{i,\geq j}$  under the constraint  $\tilde{W}_{i,< j} = \hat{W}_{i,< j} (= W'^{(j-1)}_{i,< j})$ , i.e.,

$$\mathbf{0}_{1 \times m-j+1} = \nabla_{\tilde{\mathbf{W}}_{i,\geq j}} L'_{i,\lambda}(\tilde{\mathbf{W}}_{i,:}) \Big|_{\tilde{\mathbf{W}}_{i,:}=\mathbf{W}'_{i,:}^{(j-1)}} \\ = -\Big(\mathbf{W}'^{(0)}_{i,< j} - \underbrace{\mathbf{W}'^{(j-1)}_{i,< j}}_{=\hat{\mathbf{W}}_{i,< j}}\Big) \mathbf{H}'_{< j,\geq j} - \Big(\mathbf{W}'^{(0)}_{i,\geq j} - \mathbf{W}'^{(j-1)}_{i,\geq j}\Big) \mathbf{H}'_{\geq j,\geq j} \\ = \hat{\mathbf{W}}_{i,< j} \mathbf{H}'_{< j,\geq j} + \mathbf{W}'^{(j-1)}_{i,\geq j} \mathbf{H}'_{\geq j,\geq j} - \mathbf{W}'^{(0)}_{i,:} \mathbf{H}'_{:,\geq j}.$$
(20)

Similarly, by definition in Equation 16,  $W_{i,:}^{\prime(j)}$  is also a stationary point of  $L_{i,\lambda}^{\prime}(\tilde{W}_{i,:})$ , however, this time only with respect to  $\tilde{W}_{i,>j}$  rather than  $\tilde{W}_{i,\geq j}$ , i.e., the derivative with respect to  $\tilde{W}_{ij}$  at  $W_{i,:}^{\prime(j)}$  may take some (in general) non-zero value  $\gamma_{ij}$ . It turns out that the following calculations are easier if we nevertheless include the derivative with respect to  $\tilde{W}_{ij}$ , and explicitly keep track of the fact that it is not necessarily zero. Thus,

$$(\gamma_{ij}, \mathbf{0}_{1 \times m-j}) = \nabla_{\tilde{\boldsymbol{W}}_{i, \geq j}} L'_{i, \lambda}(\tilde{\boldsymbol{W}}_{i, :}) |_{\tilde{\boldsymbol{W}}_{i, :} = \boldsymbol{W}'_{i, :}}^{(j)}$$
  
=  $\hat{\boldsymbol{W}}_{i, < j} \boldsymbol{H}'_{< j, \geq j} + \boldsymbol{W}'^{(j)}_{i, \geq j} \boldsymbol{H}'_{\geq j, \geq j} - \boldsymbol{W}'^{(0)}_{i, :} \boldsymbol{H}'_{:, \geq j}.$  (21)

Subtracting Equation 20 from Equation 21 leads to a cancellation of the terms with  $\hat{W}_{i,<j}$  and  $W'_{i,:}^{(0)}$ , and we find

$$W_{i,\geq j}^{\prime(j)}H_{\geq j,\geq j}^{\prime} = W_{i,\geq j}^{\prime(j-1)}H_{\geq j,\geq j}^{\prime} + (\gamma_{ij}, \ \mathbf{0}_{1\times m-j}).$$
(22)

We solve for  $\pmb{W}_{i,\geq j}^{\prime(j)}$  by multiplying from the right with  $(\pmb{H}_{\geq j,\geq j}^{\prime})^{-1}$ , and find

$$\boldsymbol{W}_{i,\geq j}^{\prime(j)} = \boldsymbol{W}_{i,\geq j}^{\prime(j-1)} + (\gamma_{ij}, \ \boldsymbol{0}_{1\times m-j})(\boldsymbol{H}_{\geq j,\geq j}^{\prime})^{-1} = \boldsymbol{W}_{i,\geq j}^{\prime(j-1)} + \gamma_{ij} \big[ (\boldsymbol{H}_{\geq j,\geq j}^{\prime})^{-1} \big]_{j,\geq j}.$$
 (23)

Finally, we solve for  $\gamma_{ij}$  by evaluating Equation 23 at indices (i, j), recalling that  $W'_{ij}^{(j)} = \hat{W}_{ij}$  by definition. Thus,

$$\hat{W}_{ij} = W_{ij}^{\prime(j-1)} + \gamma_{ij} \left[ (\boldsymbol{H}_{\geq j, \geq j}^{\prime})^{-1} \right]_{jj} \qquad \Longrightarrow \qquad \gamma_{ij} = \frac{\hat{W}_{ij} - W_{ij}^{\prime(j-1)}}{\left[ (\boldsymbol{H}_{\geq j, \geq j}^{\prime})^{-1} \right]_{jj}}.$$
 (24)

Inserting this result for  $\gamma_{ij}$  into Equation 23 leads to  $W'_{i,>j}^{(j)} = W'_{i,>j}^{(j-1)} + \Delta_{W'_{i,>j}}$  with  $\Delta_{W'_{i,>j}}$  as given in Equation 10 of the main text (recall that, in Equation 10,  $W'_{ij}$  refers to an unquantized weight that has already been updated in the previous iterations, see Algorithm 1; this is  $W'_{ij}^{(j-1)}$  in the more explicit notation used in this appendix).

# C.4 Proof of Proposition (ii)

For fixed  $\hat{W}_{i,<j}$ , we consider the function  $\tilde{W}_{i,\geq j} \mapsto L'_{i,\lambda}((\hat{W}_{i,<j}, \tilde{W}_{i,\geq j}))$ . Since this is a quadratic function with Hessian  $H'_{\geq j,\geq j}$  and a stationary point at  $W'_{i,\geq j}$  by Equation 18, we have

$$L_{i,\lambda}'((\hat{\boldsymbol{W}}_{i,
(25)$$

Evaluating Equation 25 at  $\tilde{W}_{i,\geq j} = W_{i,\geq j}^{\prime(j)}$  and subtracting the first term on the right-hand side of Equation 25, we thus find

$$\begin{aligned} \Delta_{L'} &= L'_{i,\lambda} \big( \boldsymbol{W}_{i,:}^{\prime(j)} \big) - L'_{i,\lambda} \big( \boldsymbol{W}_{i,:}^{\prime(j-1)} \big) \\ &= L'_{i,\lambda} \big( (\hat{\boldsymbol{W}}_{i,j}^{\prime}} \boldsymbol{H}_{\geq j,\geq j}^{\prime} (\Delta_{\boldsymbol{W}_{i,>j}^{\prime}})^{T} \end{aligned}$$
(26)

where we used that  $W_{i,>j}^{\prime(j)} - W_{i,>j}^{\prime(j-1)} = \Delta_{W_{i,>j}^{\prime}}$ . Inserting  $\Delta_{W_{i,>j}^{\prime}}$  from Equation 10, we find

$$\Delta_{L'} = \frac{1}{2} \left( \frac{W'_{ij} - \hat{W}_{i,j}}{\left[ (\boldsymbol{H}'_{\geq j,\geq j})^{-1} \right]_{jj}} \right)^2 \underbrace{\left[ (\boldsymbol{H}'_{\geq j,\geq j})^{-1} \right]_{j,>j} \boldsymbol{H}'_{\geq j,\geq j} \left[ (\boldsymbol{H}'_{\geq j,\geq j})^{-1} \right]_{j,j}}_{= \left[ (\boldsymbol{H}'_{\geq j,\geq j})^{-1} \right]_{jj}} = \frac{1}{2} \frac{(W'_{ij} - \hat{W}_{i,j})^2}{\left[ (\boldsymbol{H}'_{\geq j,\geq j})^{-1} \right]_{jj}}$$
(27)

as claimed in the second part of Equation 10.

### C.5 Proof of Proposition (iii)

To proof equivalence of Equation 10 and Equation 11, we have to show that, for all  $k \ge j$ ,

$$\frac{[(\boldsymbol{H}'_{\geq j,\geq j})^{-1}]_{jk}}{[(\boldsymbol{H}'_{\geq j,\geq j})^{-1}]_{jj}} = \frac{C'_{jk}}{C'_{jj}} \quad \text{and} \quad [(\boldsymbol{H}'_{\geq j,\geq j})^{-1}]_{jj} = (C'_{jj})^2$$
(28)

where C' is an upper triangular matrix that satisfies  $C'^T C' = (H')^{-1}$ . Using block-matrix notation,

$$C' = \begin{pmatrix} C'_{\langle j, \langle j \rangle} & C'_{\langle j, \geq j} \\ 0 & C'_{\geq j, \geq j} \end{pmatrix},$$
(29)

it is easy to verify (by multiplying out  $(C')^{-1}C'$  and verifying that the result is the identity) that

$$(\mathbf{C}')^{-1} = \begin{pmatrix} (\mathbf{C}'_{ (30)$$

Thus, the block form of the relation  $H' = (C'^T C')^{-1} = (C')^{-1} (C'^T)^{-1} \equiv (C')^{-1} (C')^{-T}$  is

$$\begin{pmatrix} \mathbf{H}'_{\langle j, \langle j \ } & \mathbf{H}'_{\langle j, \geq j} \\ \mathbf{H}'_{\geq j, \langle j \ } & \mathbf{H}'_{\geq j, \geq j} \end{pmatrix} = \begin{pmatrix} \star & \star \\ \mathbf{0} & (\mathbf{C}'_{\geq j, \geq j})^{-1} \end{pmatrix} \begin{pmatrix} \star & \mathbf{0} \\ \star & (\mathbf{C}'_{\geq j, \geq j})^{-T} \end{pmatrix}$$
(31)

where " $\star$ " denotes terms that are irrelevant for our proof. We can read off  $H'_{\geq j,\geq j}$  from Equation 31,

$$H'_{\geq j,\geq j} = (C'_{\geq j,\geq j})^{-1} (C'_{\geq j,\geq j})^{-T} \implies (H'_{\geq j,\geq j})^{-1} = (C'_{\geq j,\geq j})^{T} C'_{\geq j,\geq j}.$$
 (32)

Thus, by explicitly evaluating the (j,k)-th entry of  $(\mathbf{H}'_{\geq j,\geq j})^{-1}$ , we find for all  $k \geq j$ ,

$$\left[ (\mathbf{H}_{\geq j,\geq j}')^{-1} \right]_{jk} = \sum_{l=j}^{m} C_{lj}' C_{lk}' = C_{jj}' C_{jk}'$$
(33)

where only the term with l = j contributes to the sum because  $C'_{lj} = 0$  for l > j since C' is upper triangular. Both parts of Equation 28 follow immediately from Equation 33.