Quantority: Parameter Prioritization for Incremental Updates of Convolutional Neural Networks in Small Satellite Missions

Olga Kondrateva*, Stefan Dietzel† and Björn Scheuermann‡
*Technical University of Darmstadt, Darmstadt, Germany, olga.kondrateva@kom.tu-darmstadt.de
†Merantix Momentum GmbH, Berlin, Germany, stefan@merantix-momentum.com
‡Technical University of Darmstadt, Darmstadt, Germany, scheuermann@kom.tu-darmstadt.de

***Abstract*—Earth observation using small satellites operating in low Earth orbit (LEO) has received increasing interest over the past years. With high-resolution sensors amassing up to terabytes of data per day in large satellite constellations, onboard processing using neural network models is considered to manage downlink communication capacity. A question frequently overlooked, however, is how to update these models using the – often orders of magnitude smaller – uplink capacity. Yet, model updates are crucial to allow for flexible long-term missions and commercial Earth-observation-as-a-service models. In this paper, we propose an efficient communication protocol for model updates based on incremental transmission of prioritized parameters and vector quantization. The evaluation results show that our approach significantly outperforms existing incremental model update transmission schemes.**

I. Introduction

Recently, the number of satellite missions to support Earth observation tasks has increased significantly. A major enabler are technological advancements that allow for much smaller cost-effective satellites containing commercial off-the-shelf components instead of more expensive radiation-hardened electronics. A popular example are CubeSats [1], which comprise $10\times10\times10$ cm units, operate in LEO, and are often used in larger formations. At the same time, sensor technology used in satellites has evolved massively. In many cases, both high spacial and spectral resolution are required to meet the application demands. CubeSats can be equipped with multi-spectral or even hyper-spectral sensors [2], [3]. Such satellite missions can produce up to several terabytes of data per day.

Vast amounts of data necessitate onboard pre-processing before transmission to match the available downstream bandwidth. For CubeSats, for instance, downlink speeds of hundreds of Mbps up to several Gbps are feasible using an X band transmitter [4], [5]. In contrast to geostationary satellites, which appear motionless at a fixed position in the sky to ground observers, CubeSats, however, move rapidly relative to the ground station positions. Hence, communication protocols need to cope with limited communication windows of only a few minutes several times per day [6]. Due to their prohibitive cost, increasing the number of ground stations to compensate for these short contact times is not a viable strategy for most small satellite missions [6].

To cope with communication requirements, neural networks were proposed for small satellites due to their ability to process and filter data with high accuracy and efficiency [7], [8], [9]. Only the most relevant information is then transmitted downstream. Besides, neural networks can provide benefits for systems related to the satellites' operation [7], [8], [10], [11]. Several studies prove the feasibility of running neural networks onboard of small satellites [12], [9], [10].

While onboard processing using neural networks helps to compensate for communication constraints on the downlink side, it introduces new challenges in the uplink direction [13], [14]. Before launch, the initial model to be used can be preloaded on the satellites. However, many LEO missions will require recurrent updates of the neural network models when the satellites are already in orbit. The main reasons are changes in detection tasks during the satellites' mission and limited availability of training data during initial deployment when a new sensor technology is used [15], [12]. Unlike the comparatively high downlink data rates, many small satellite missions use the S band for their uplink, which is limited to a few hundreds of kbps [4], [5]. Hence, updating a model in a naïve way can take hours or even days. For instance, our simulation results show that a full transmission of VGG16, which contains around 15 million parameters, takes about 53 hours. Pauses between contacts of satellites with ground stations contribute massively to this duration, increasing the effective transmission time by a factor of 2 when compared to continuous transmission.

To cope with communication challenges of small satellite networks, we propose a flexible transmission protocol allowing for incremental updates of neural network parameters. Using our protocol, reasonable performance can be achieved even with partially updated models. The simulation results demonstrate that our approach considerably outperforms existing transmission schemes that incorporate incremental updates.

Many studies in the field of model compression show that model parameters are not equally important [16], [17]. Hence, prioritizing more important parameters for transmission leads to more rapid improvements in accuracy. However, changing the parameters' transmission order involves a considerable

ISBN 978-3-903176-63-8 ©2024 IFIP

overhead, since this information is needed to reconstruct the network onboard the satellite. For instance, the naïve method of communicating the parameter order directly would require $N \log_2 N$ bits for N parameters with $\log_2 N$ bits needed to encode each position. Moreover, Shannon's theory [18] suggests that $\mathcal{O}(\log_2 N!)$ bits are a lower bound for a lossless encoding of such a permutation. Therefore, finding good approximations is crucial to make prioritization approaches practically applicable.

In our protocol, we leverage parameter prioritization while reducing the overhead to N bits. We separate parameters into a prioritized and a non-prioritized group based on their absolute values as a simple yet effective criterion. Within each group, however, we keep the original parameter order for transmission. Since the original order is assumed to be known by both the sender and the receiver, we thereby reduce the overhead to one bit per parameter, which is used to indicate its priority status. Neural networks typically contain many parameters with near-zero values due to regularization techniques, such as L_1 or L_2 regularization, which penalize large values during training [16], [19]. Therefore, a relatively small subset of prioritized weights with all other weights set to zero suffices to approximate the updated model. To further reduce the transmission time for this initial approximation, we apply vector quantization, which provides a lossy compression of the prioritized parameters by approximating them using entries from a model-specific codebook. To achieve compression, the parameters, grouped into vectors, are replaced by indices from the codebook.

In total, our approach thus comprises three stages: First, we transmit the quantized version of the prioritized weights, represented as the codebook and the index structure, together with the bit array indicating the prioritized parameter positions. Second, we incrementally transmit the exact prioritized weights, gradually improving the model's accuracy. Finally, all other exact weights follow until the complete updated model is available at the satellite. Thereby, we drastically reduce the transmission time for a first usable approximation before incrementally updating the network until it is fully transmitted.

Our contributions can be summarized as follows:

- We propose to use parameter prioritization and vector quantization to enable incremental updates of neural network parameters.
- We propose an algorithm to efficiently find the best tradeoff between communication overhead and accuracy.
- We extensively evaluate the effects of different parameter choices and use network simulations to demonstrate that our approach considerably outperforms existing methods.

The remainder of this paper is organized as follows. In Section II, we summarize existing work with different optimization goals before providing an overview of our system model in Section III. We provide a detailed explanation of our approach in Section IV. In Section V, we evaluate our approach using a range of common neural network models, and we summarize our conclusions in Section VI.

II. Related Work

A large body of work has been devoted to finding efficient representations of neural network parameters, which can be in the order of millions or even billions. In particular, various techniques for model compression have been proposed. Examples include parameter pruning [16], [17], quantization [20], [21], low-rank decomposition [22], [23], and knowledge distillation [24], [25]. These approaches focus on reducing a network's memory footprint or on speeding up computation, whereas we consider the transmission of networks in a challenging environment. Therefore, we can apply some of the concepts proposed, namely parameter pruning and vector quantization, but extend them to design an incremental transmission scheme optimized for satellite settings.

Parameter pruning refers to removing the least significant parameters from a neural network to reduce its size or to increase computation efficiency. Various criteria were investigated to assess parameter importance, the most important being magnitude [26] and gradient magnitude [27], as well as L_1 and L_2 norms [28]. In our work, we apply the parameter magnitude criterion, which considers parameters with higher absolute value as more important. Though simple, this criterion achieves excellent results [26] and is still used in recent works on pruning [29]. Parameter pruning is an iterative approach: In each step, a small set of least important parameters is removed, and the network is retrained to restore its accuracy. Our approach, in contrast, focuses on the most important parameters to prioritize them for transmission. Since the remaining parameters are transmitted at a later stage, no retraining is necessary.

Both scalar and vector quantization [20], [21] have been successfully applied for neural network compression. Similar to parameter pruning, these are iterative techniques. In each step, only a fraction of the parameters is quantized and the non-quantized parts of the neural network are retrained to restore the accuracy. Unlike the standard approach, we quantize the parameters in a single step and do not apply retraining. A further important difference concerns the number of codebooks. While using vector quantization as a standalone approach for compression requires multiple codebooks to achieve competitive accuracy levels, we use only a single codebook. Employing vector quantization as part of our incremental transmission scheme, we have a mixture of quantized and original values available at the satellite at each step. This allows us to considerably reduce the overhead for quantization.

Several works consider various aspects of updating model parameters over wireless networks, making them relevant to our setting. The approach proposed in [14] considers parameter prioritization and is based on a lossy permutation compression technique called "sorting subsequences" [30]. Parameter prioritization is achieved by introducing multiple equally-sized priority groups, which are then incrementally transmitted. The tradeoff between the rate of accuracy improvement and the incurred overhead is controlled by choosing the number of priority groups. The main difference to our approach is that, instead of increasing the number of priority groups, our goal is to prioritize as few parameters as possible. This way, it can be guaranteed that the most important parameters arrive as soon as possible, and the transmission overhead can be reduced considerably. Furthermore, Jankowski et al. [31] propose to use noise injection during the training of neural networks. This approach increases the robustness of neural networks in the presence of channel noise. In contrast to our mechanism, this work does not consider disruption-tolerant networking and partial availability of neural network parameters. Finally, Kondrateva et al. [13] apply vector quantization to create an approximation of neural network parameters that serves as basis for subsequent transmission of exact parameter values. As opposed to the approach presented here, this work does not consider parameter prioritization and quantizes all parameters. We quantize only the prioritized parameters, which allows us to considerably reduce the overhead, significantly outperforming the purely quantization-based approach.

III. System Model

We consider an Earth observation task that is performed by one or more satellites operating in LEO. In order to reduce communication overhead, we assume that a neural network model is used onboard the satellites to perform a classification task and only communicate the most relevant observations to Earth. As requirements may change during the satellite mission, periodic model updates are needed. These updates either change the classification task completely, or they provide an updated model to improve performance for the original task. Whenever an update is required, a new model is trained on Earth and its updated parameters (i. e., model weights) need to be transmitted to the satellites.

For typical satellite use cases, the model architecture can be assumed stable and to be known by the satellites. Therefore, we regard the updated model weights as a flattened vector

$$W = [w_1, \dots, w_N]. \tag{1}$$

Both the ground station and the satellites use a deterministic mapping function to retrieve the flattened weights vector given a neural network model data structure and vice versa.

The goal of our approach is to make the update process – that is, transmitting the updated weights vector W – as efficient as possible in order to lessen the time required until the new model is operational on the satellite.

IV. Incremental Updates of Neural Networks

Figure 1 shows an overview of our transmission scheme. We start with the flattened vector of updated weights W (1). First, we prioritize a subset of weights based on their absolute value (2), as larger weights have more impact on the model's performance. To avoid having to encode and transmit the permutation that determines the sort order, we use an approximate prioritization technique. That is, we transmit prioritized weights first but use the original order from the array W within that subset. This order is assumed to be known at the satellite, since the model architecture remains unchanged. That way, a compact bit array B (3) that denotes which weights are part of the prioritized subset W_p suffices for reconstruction of the weights' position in the neural network structure on the satellite. The fraction of prioritized weights is chosen based on the minimum required model performance, e. g., minimum accuracy, which is determined by the use case.

To further improve the accuracy early on, we calculate a codebook using vector quantization and quantize the prioritized weights as W_p^* (4), which are represented using a list of indices from the codebook C, called the index structure X. We then transmit the bit array B to the satellite, followed by the codebook C and index structure X of the quantized weight vectors (5). The exact prioritized weights are then incrementally transmitted in the order indicated by the bit array (6), followed by all remaining weights (7).

The satellite first receives the bit array B, the codebook C, and the index structure X. These data structures are used to initiate the updated network with the quantized values from W_p^*; the remaining values are set to zero (8). The underlying idea is that the data structures required to reconstruct this quantized, prioritized subset of weights can often be transmitted within a single communication window between the ground station and the satellite. When the original weights start to arrive, the previous values – which are either quantized values or zeros – are gradually replaced, and the accuracy increases further (9) until the fully updated network is reconstructed (10). To avoid frequent, costly model updates on the satellite, each transmission contains a flag bit, indicating whether the newly received weights should be buffered or used immediately. That way, the satellite only updates the model when a sufficient increase in accuracy can be expected. Suitable points for model updates can be precomputed on Earth. Together, both the prioritization and the vector quantization help to have an operational, high accuracy representation of the updated model available at the satellite as early as possible.

A. Parameter prioritization

Parameter prioritization is an effective way to significantly lower the time until a partially transmitted neural network model achieves a good accuracy. Its effectiveness hinges on the observation that parameters with higher absolute value have a much larger impact on the models' performance. To determine the best fraction of weights to prioritize, we search for a cutoff point that achieves sufficient accuracy for a given use case with as few prioritized weights as possible. As the accuracy does not necessarily increase monotonously with the fraction of prioritized weights, we introduce a recursive binary search approach that avoids local optima.

The ideal cutoff point k is determined based on two application-dependent parameters. The minimum accuracy level a_{min} determines the minimum model performance that leads to usable results for a given application. In addition, we introduce w_{sat} to capture how quickly the desired accuracy should be achieved. More specifically, w_{sat} denotes the fraction of available exact weights at the satellite with which the approximated model should achieve a_{min}. For example, the

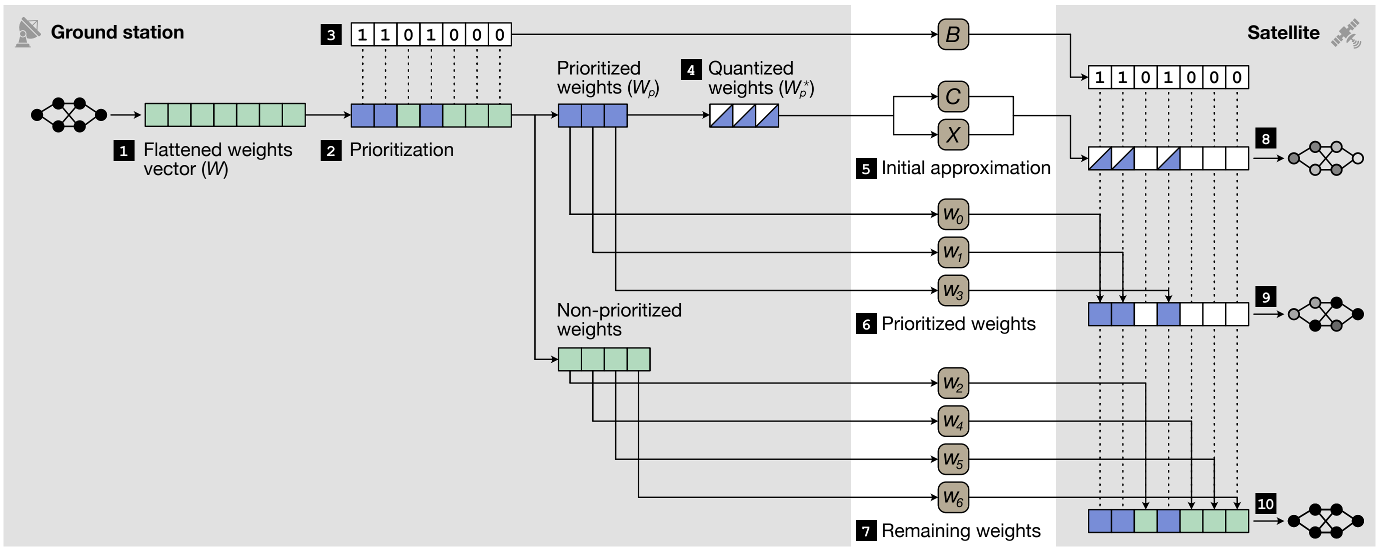


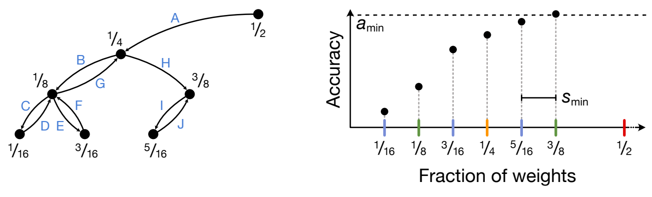
Fig. 1: Overview of processing and transmission steps.

goal could be to achieve 90% accuracy after transmitting 10% of the original parameters. We also define the minimum step size $s_{\min}$ to be used as stop criterion.

```
   Best-Split ($s, e, W, a_{\min}, w_{\text{sat}}, s_{\min}$)
1  if $e - s < s_{\min}$ then
2      return 0, 0;
3  $m = \frac{e-s}{2}$;
4  $a_{\text{left}}, k_{\text{left}}$ = Best-Split ($s, m, W, a_{\min}, w_{\text{sat}}, s_{\min}$);
5  if $a_{\text{left}} \geq a_{\min}$ then
6      return $a_{\text{left}}, k_{\text{left}}$;
7  $a_m$ = Evaluate ($W, m, w_{\text{sat}}$);
8  if $a_m \geq a_{\min}$ then
9      return $a_m, m$
10 $a_{\text{right}}, k_{\text{right}}$ =
     Best-Split ($m, e, W, a_{\min}, w_{\text{sat}}, s_{\min}$);
11 if $a_{\text{right}} \geq a_{\min}$ then
12     return $a_{\text{right}}, k_{\text{right}}$;
13 else
14     return Best (($a_{\text{left}}, k_{\text{left}}$), ($a_m, m$), ($a_{\text{right}}, k_{\text{right}}$));
```

Algorithm 1: Determining the best cutoff point.

Algorithm 1 shows the binary search approach. The goal is to find the minimum subset W_p of the weights W that should be prioritized to achieve a performance of $a_{\min}$ or better with w_{sat} percent of the available original weights. As additional stop criterion, we provide a minimum interval size $s_{\min}$, and W represents the weight vector (cf. Equation (1)). We start with the whole interval $[0, 1]$ (i. e., $s = 0$ and $e = 1$); that is, we regard the whole weights vector W, sorted by absolute value. Line 2 is the stop criterion: The algorithm terminates if the current interval becomes smaller than $s_{\min}$. Otherwise, the current interval is split in the middle m (Line 3). That is, the first candidate cutoff point m considers the first half of the sorted weights vector as prioritized. Since we seek to minimize the number of prioritized weights, the algorithm first recurses into the left half and determines the best accuracy a_{left} and corresponding cutoff point k_{left} (Line 4). If the result fulfills the requirement $a_{\min}$, the recursion terminates. Otherwise, the accuracy at the middle point m is determined (Line 7). The function `Evaluate` simulates the execution of the transmission protocol up to the point where w_{sat} percent of the original weights are transmitted and using m as the candidate cutoff point. The partially transmitted model's accuracy is then evaluated and returned as a_m. If the minimum desired accuracy is still not achieved, the algorithm recurses into the right half of the interval (Line 10) and returns the determined cutoff point k_{right}. If all cutoff points fail to meet $a_{\min}$, the best cutoff point found is returned (Line 14) to guarantee that the algorithm terminates even if the desired parameter combination $a_{\min}$ and w_{sat} cannot be achieved. Since the algorithm always checks the left side first, the cutoff point is minimal under the chosen criteria.



(a) Recursion (b) Accuracy

Fig. 2: Recursive binary search for an example network.

Figure 2 shows the cutoff point computation for the LeNet5 model [32] trained on MNIST dataset [33] with $a_{\min} = 0.95$, $s_{\min} = 1/16$, and $w_{\text{sat}} = 0.1$. The algorithm first recurses to the left (Steps A, B, C) until $s_{\min}$ is reached, as shown in Figure 2a. As $a_{\min}$ is not reached with a cutoff point at $1/16$ (see Figure 2b), the recursion backtracks (Step D) and evaluates the accuracy for $1/8$, still not reaching the minimum threshold.

Next, the right branch is evaluated (Step E) and returns (Step F). The algorithm continues in a similar fashion: backtracking to $1/4$ (Step G) and then recursing into the right branch next (Step H). From there, the recursion again first continues left (Step I), failing to meet the minimum accuracy criterion at $5/16$. Finally, $3/8$ is evaluated as candidate cutoff point (Step J), which meets the minimum accuracy of $a_{\min} = 0.95$. That is, prioritizing the largest by absolute value $3/8$ of all weights fulfills the minimum accuracy requirement.

After the right fraction of prioritized weights k has been determined, it is communicated to the satellite. For this, we use a bit array that indicates which parameters are prioritized:

$$B = [b_1, \ldots, b_N] \,, \tag{2}$$

where $b_i = 1$ iff w_i is prioritized. Accordingly, we define the set of prioritized weights W_p as

$$W_p = [w_i \in W \colon b_i = 1] \,. \tag{3}$$

Within W_p, we keep the original parameter order of W. Thereby, we prioritize the most important weights, yet only require the compact bit array B for determining the weights' position within the neural network structure.

B. *Vector quantization*

We apply vector quantization to the prioritized weights W_p in order to more quickly convey a first approximation during the early transmission stages. During later stages, quantized weights are replaced by the exact weights, as more and more transmissions arrive at the satellite. Vector quantization generalizes the idea of scalar quantization, i. e., mapping exact values to a smaller set of approximate values called a codebook. In vector quantization, data vectors $v \in \mathbb{R}^D$ are assigned to the nearest centroids $c_i \in \mathbb{R}^D, i \in \{1, \ldots, K\}$ from a codebook $C = [c_1, \ldots, c_K]$ using some distortion measure, e. g., mean squared error. The codebook size K and vector length D are design parameters. Vector quantization allows to achieve higher compression results, since apart from the codebook size K, the compression rate can be controlled by the vector length D. In our protocol, we determine the best parameter combination of K and D as part of the `Evaluate` function in Algorithm 1. That is, as explained in Section IV-A, the protocol execution is simulated for a range of values for K and D, and the best combination is selected.

The encoding algorithm is implemented as follows. First, the weights W_p are partitioned into vectors of length D, w. l. o. g. assuming that $|W_p|$ is divisible by D:

$$V = \big[[w_1, \ldots, w_D],\ [w_{D+1}, \ldots, w_{2D}],\ \ldots\big] \tag{4}$$

Then we apply K-means clustering to partition V into K disjoint sets $\{S_1, \ldots, S_K\}$ by choosing for each $v \in V$ the set S_i with the minimal distance to its centroid c_i. The centroids c_i are computed as the mean of all vectors $v \in S_i$. That is, we iteratively obtain the best partition as follows:

$$\min \sum_{k=1}^{K} \sum_{v \in S_k} \left\| v - c_k \right\|_2^2 . \tag{5}$$

As the result of these computations, we obtain the codebook C containing the centroids:

$$C = [c_1, \ldots, c_K]. \tag{6}$$

Using the codebook and the set of vectors V, we can calculate the index structure to approximate the prioritized weight vector W_p as W_p^*:

$$X = [x_1, \ldots, x_{|W_p|/D} \colon x_i = \sigma(v_i)], \tag{7}$$

where $\sigma(v_i)$ is the index of v_i's nearest centroid $c_j \in C$.

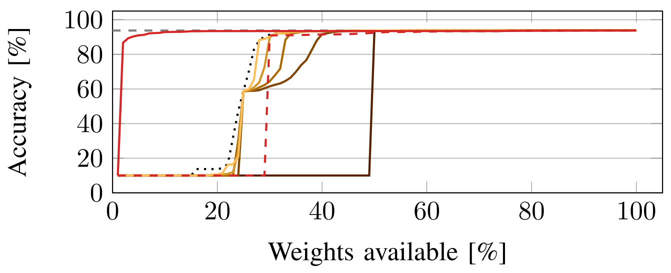
The index structure X is then transmitted to the satellite, together with the codebook C and the bit vector B. At the satellite, an approximation W_p^* can be reconstructed using these data structures by replacing the index values x_i with the corresponding centroids c_j and putting them at the positions set to 1 in B. After this initial reconstruction, the exact prioritized weights $w_i \in W_p$ are incrementally transmitted to the satellite, followed by the non-prioritized weights.

V. Evaluation

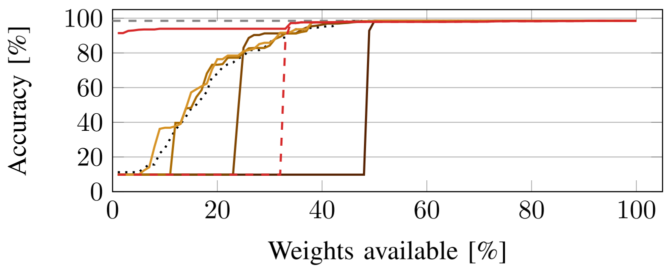
We evaluate the performance of our approach based on several metrics. First, in Section V-A, we compare our approach with and without using vector quantization to the fully prioritized parameter order and to an existing scheme for parameter prioritization proposed in [14]. Second, we show that the results achieved substantially outperform results of an existing vector quantization scheme in Section V-B. Third, in Section V-C, we consider the overhead incurred by our approach and compare it with other transmission schemes presented in [13] and [14]. Finally, we evaluate our approach with the help of a network simulator in Section V-D.

A. *Optimizing parameter order*

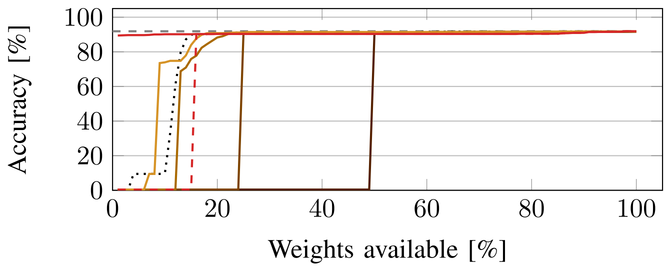
We compare our transmission protocol to the fully prioritized transmission order, i. e., all weights sorted by absolute value. In addition, we evaluate the protocol proposed in [14] (hereafter: $\mathcal{EG}$), which uses equally sized priority groups (see Section II). To assess how soon the model can become operational at the satellite, we evaluate how often the model's prediction is exactly the expected result (top-1 accuracy) depending on the amount of neural network parameters available at the satellite. We regard the accuracy in isolation and defer the discussion of overhead to later subsections. First, consider the results for a VGG16 neural network [35] trained on the CIFAR-10 [36] dataset (Figure 3a). The x-axis shows the amount of weights transmitted to the satellite, and the y-axis shows the resulting model accuracy. When the weights are transmitted in strictly prioritized order, at least 29% of parameters are necessary to achieve $> 90\%$ accuracy. The transmission can be roughly divided into three stages based on the resulting plot's shape. At first, the accuracy does not improve despite the increasing number of available parameters (1–15%). During the second stage, the accuracy improves rapidly and quickly achieves near optimal values. During the third stage, the accuracy only improves marginally. Therefore, the optimal parameter order is not essential at least during the first and the third stages,



(a) VGG16 trained on CIFAR10.



(b) LeNet5 trained on MNIST.



(c) Makantasis et al.'s model [34] trained on Indian Pines dataset.

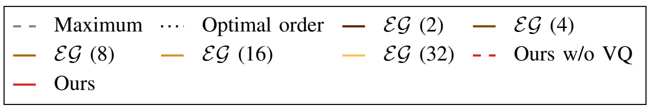


Fig. 3: Comparison of parameter prioritization techniques.

which motivates our approach of transmitting in approximately prioritized order.

For our approach, we show the results with and without vector quantization. We set the minimum accuracy $a_{\text{min}} = 92\%$, step size $s_{\text{min}} = 2$, and the amount of exact parameter values w_{sat} needed to achieve a_{min} to 10%. This configuration results in prioritizing 29.7% of all parameters. This value is very close to 29%, the first value for which a high accuracy is achieved in the case of the optimal parameter order. The results show that when 30% of weights become available – that is, when all prioritized parameters are transmitted – the optimal parameter order and our approach perform similarly. Obviously, it would be possible to improve the accuracy earlier by prioritizing less weights. However, it would take longer to achieve higher accuracies, since, in this case, some of the important but not prioritized parameters would be transmitted later.

Next, we evaluate the impact of vector quantization with a codebook size of 128 and a vector length of 4, as computed by Algorithm 1. The results show that vector quantization allows to achieve high accuracies much earlier, because quantization allows to transmit the initial approximation of the prioritized weights more quickly. Whereas approximately 48% of original parameter values are required to achieve a_{min} without vector quantization, 92.12% accuracy is achieved with only 7% of the available weights when applying vector quantization. Note that a considerable improvement in accuracy of 86.69% is attained with only 2% of parameters.

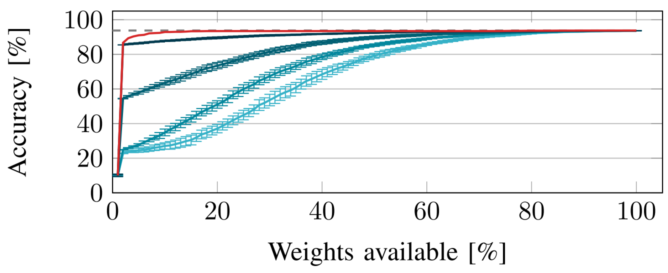
For $\mathcal{EG}$, the results improve as the number of priority groups increases. However, the performance only comes close to the strictly prioritized order when using a large number of groups (32). In this case, $\log_2 32 = 5$ bits are required to encode the priority group for each parameter, while our approach (without vector quantization) achieves the same accuracy with a 5 times lower overhead. When vector quantization is applied, our approach considerably outperforms $\mathcal{EG}$.

Figure 3b shows the results of LeNet5 [32] trained on MNIST [33]. Both the model architecture and the dataset used for training are much simpler compared to the previous example. For our approach, we compute the amount of prioritized parameters as 33.59% using the same parameters as before. For this model, the first phase is very short and the accuracy starts to improve quickly. During the second phase, the exact parameter order is important, explaining why our approach achieves lower accuracy levels during early transmission stages when used without vector quantization. Nevertheless, our results still considerably outperform $\mathcal{EG}$ for two priority groups while incurring the same overhead. Similar to the previous model, the performance of $\mathcal{EG}$ improves with more priority groups. For 16 groups it even becomes slightly better than in the case of the fully prioritized order. However, choosing even higher group numbers does not lead to further improvement and incurs a prohibitive overhead. This is in contrast to our approach, which allows to achieve much better accuracy when vector quantization is applied on top of parameter prioritization.

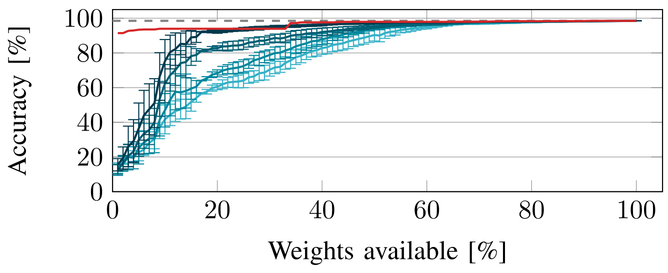
To demonstrate the applicability of our approach to satellite imagery, we evaluate it on a neural network proposed by Makantasis et al. [34]. The model is trained on the Indian pines dataset [37] containing hyperspectral images with a total of 220 bands, which we reduce to 80 bands using principal component analysis. Figure 3c shows the results. As the model achieves a maximum accuracy of 91.88%, we set $a_{\text{min}} = 90\%$, resulting in 15.63% of prioritized weights according to Algorithm 1. The other parameters remain unchanged. For the strictly prioritized transmission order, the accuracy is within 1% of the maximum when 22% of weights are available. As in previous examples, the performance of $\mathcal{EG}$ improves with the number of priority groups. Our approach without vector quantization achieves the accuracy of 90.28% when the prioritized parameters become available. The a_{min} accuracy of 90% with only 8% of original weights is reached by our approach when vector quantization with parameters $4/64$ is applied.

B. Vector quantization

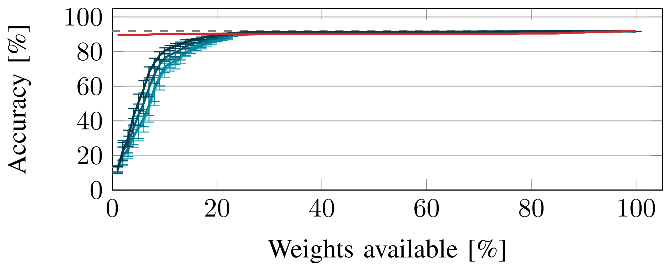
We now compare the proposed approach to the transmission scheme presented in [13] (hereafter: $\mathcal{VQ}$). Similar to our



(a) VGG16 trained on CIFAR-10.



(b) LeNet5 trained on MNIST.



(c) Makantasis et al.'s model [34] trained on Indian Pines dataset.

Fig. 4: Comparison of vector quantization techniques.

approach, $\mathcal{VQ}$ applies vector quantization (see Section II). Since $\mathcal{VQ}$ transmits weights in random order, we evaluate it ten times for each combination. Error bars show 95% confidence intervals. Figure 4a presents the results for the VGG16 model. We use a vector length of 4 and a codebook size of 128 for our approach, as computed by Algorithm 1. For $\mathcal{VQ}$, we use a vector length of 4 and different codebook sizes, since the accuracy achieved with these combinations is closest to ours. The results of $\mathcal{VQ}$ highly depend on the chosen codebook size. The most direct comparison is to $\mathcal{VQ}$ $(4/128)$, because the same parameters are used by our algorithm. Evidently, our approach drastically outperforms $\mathcal{VQ}$. While our transmission scheme achieves 92.9% accuracy with 10% of the original weights, $\mathcal{VQ}$ reaches only $34.9 \pm 1.9\%$. Though the gap becomes smaller as more weights become available, $> 92.9\%$ accuracy is reached by $\mathcal{VQ}$ only with 83% of the weights. With $\mathcal{VQ}$ $(4/512)$, which achieves the best accuracy among all combinations, $87.8 \pm 0.3\%$ can be attained with 10% of the exact weights, which is still 5.1% lower than our result. Both approaches perform similarly only for $> 50\%$ of available weights.

The results for the LeNet5 model are shown in Figure 4b. We use the combination $4/64$ for our approach. Similar to the previous example, we considerably outperform the results both for $\mathcal{VQ}$ $(4/64)$ (93.9% vs. $41.7 \pm 8.3\%$ for 10% weight availability) and $\mathcal{VQ}$ $(4/512)$ (93.6% vs. $75.2 \pm 12.5\%$ for 10% weight availability). Note that the error bars for $\mathcal{VQ}$ are much larger than in case of the VGG16 model. This behavior is due to the weights being transmitted in random order. Since the LeNet5 model is much smaller than VGG16, it is more sensitive to the order in which parameters are transmitted. Our approach, in contrast, involves no randomness and is hence more stable especially for small models.

Finally, we show the results for Makantasis et al.'s model in Figure 4c. Although $\mathcal{VQ}$ already achieves quite good performance, our approach significantly improves the accuracy during the early transmission stages. In particular, our approach outperforms the best result of $\mathcal{VQ}$ $(4/512)$ by about 10% with 10% of the original weights.

C. Evaluating the overhead

Next, we consider the overhead incurred by our approach, which is determined by the sizes of the bit vector B, the codebook C, and the index structure X:

$$\underbrace{N}_{B} + \underbrace{KD}_{C} + \underbrace{\lceil NkD^{-1} \rceil \log_2 K}_{X}. \tag{8}$$

One bit per parameter, i. e., N bits in total, is used to encode B. The size of the codebook C and index structure X both depend on the vector length D and the number of the codebook entries K (see Section IV-B). Since C contains K entries of dimension D, its size is KD. As only prioritized parameters are quantized, the size of the index structure is determined by the number of parameters N multiplied by the cutoff point k and divided by the vector length D to compute the number of entries; $\log_2 K$ bits are used to encode each entry.

The overhead for $\mathcal{EG}$ assuming G groups is:

$$N \lceil \log_2 G \rceil, \tag{9}$$

where $\log_2 G$ is the number of bits needed to encode the group indices. This corresponds to the first component in Equation (8). As we do not consider multiple priority groups, our overhead is one bit per weight, equal to $\mathcal{EG}$ with $G = 2$.

The overhead for $\mathcal{VQ}$ is:

$$\lceil ND^{-1} \rceil \log_2 K. \tag{10}$$

This corresponds to the third component in Equation (8). Since $\mathcal{VQ}$ assumes that a single codebook is reused, the second component is not present. However, our approach only quantizes Nk weights, where k is the fraction determined by our binary search algorithm and ≈ 0.3 in practice.

To determine the impact of the different overheads, we relate it to the achieved accuracy for the VGG16 model containing 15 million parameters. The results are presented in Table I. We compare the accuracy for three snapshots, representing 10, 20, and 30 percent of all parameters available at the satellite. For

TABLE I: Overhead

Algorithm	Accuracy for different snapshots [%] 10%	20%	30%	Size [MB]
$\mathcal{VQ}$ (4/512)	87.8 ± 0.3	89.6 ± 0.5	90.9 ± 0.4	4.14
$\mathcal{VQ}$ (4/64)	27 ± 0.1	37 ± 2.4	53.6 ± 3.2	**2.76**
$\mathcal{EG}$ (4)	10	10	61.5	3.75
$\mathcal{EG}$ (32)	10	10.52	90.2	9.38
Ours (4/128)	**92.9**	**93.4**	**93.4**	2.85

our approach, we use the values determined by Algorithm 1: $D = 4$, $K = 128$, $k = 0.297$. We compare our approach to $\mathcal{EG}$ and $\mathcal{VQ}$ using two parameter combinations: the one that achieves the highest accuracy and the one that achieves the lowest overhead while showing at least some accuracy improvements for the chosen snapshots.

Our approach reaches the highest accuracy for all three snapshots and the second lowest overhead among all considered transmission schemes. The next best result is achieved by $\mathcal{VQ}$ $(4/512)$. The difference in accuracy ranges between 5.1% and 2.5%, depending on the available fraction of parameters. Furthermore, this parameter combination requires 4.14 MB to encode the additional data, while our approach only needs 2.85 MB. Thus, the proposed approach achieves considerably better accuracy and incurs 31% less overhead.

The third best performance shows $\mathcal{EG}$ with 32 groups. However, the accuracy remains low for the snapshots at 10% and 20%. This means that the model only becomes operational when at least 30% of the original parameters are transmitted. In addition, this approach incurs a prohibitive overhead of 9.38 MB, which is three times higher than that of our approach.

Finally, we consider $\mathcal{EG}$ with 4 groups and $\mathcal{VQ}$ $(4/64)$. The results show that $\mathcal{EG}$ performs considerably worse than our approach both in terms of accuracy and overhead, even when a smaller group size is chosen. Albeit $\mathcal{VQ}$ requires marginally less metadata, it reaches only $53.6 \pm 3.2\%$ accuracy for the 30% snapshot. Thus, our approach incurs the least overhead among all approaches that achieve reasonable accuracy.

D. Simulation results

To evaluate our approach in a simulated satellite communication setting, we use EstNet [38], which is based on OMNeT++ [39]. Our setup consists of a single three-unit small satellite and one ground station located in Würzburg, Germany. We do not consider constellations of satellites or inter-satellite links. We set the orbit height to 600 km and assume that the ground station communicates with the satellite using an S band antenna with a 2150 MHz carrier frequency and a receive channel bandwidth of 750 kHz. The transmission rate is set to 500 kbit/s; the satellite and ground station gains are set to 6 dBi and 18.9 dBi, respectively. The ground station transmission RF power is set to 50 W. The simulator dynamically calculates the expected signal-to-noise ratio (SNR) for the communication channel depending on the satellite's elevation angle using the simplifed general perturbation 4 (SGP4) model. We assume that updates are transmitted in

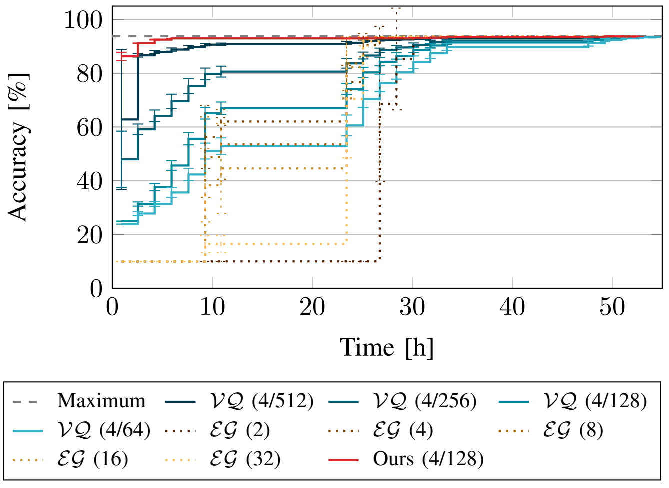


Fig. 5: Simulation results for VGG16 trained on CIFAR-10.

packets of 200 bytes and are acknowledged by the satellite. Timeouts are used for retransmission of the packets in the case of missing acknowledgements. We simulate the transmission for 53 hours. We again compare our results to $\mathcal{EG}$ (with different group sizes) and $\mathcal{VQ}$ (with different codebook sizes and vector length 4). The accuracy is evaluated at the end of each contact window. Each simulation is run ten times with different seeds. The error bars present 95% confidence intervals.

We present the results for the VGG16 model in Figure 5. The x-axis shows the transmission time and the y-axis the corresponding accuracy. The accuracy is significantly influenced by the contact window pattern. In our case, a series of short contact periods (from 1 hour to 10.88 hours and from 23.43 hours to 35.03 hours) is followed by long periods (12.5 hours) without connectivity, showing that a compact model approximation is essential. The early transmission stages are the most important, since the accuracy highly depends on the availability of metadata and some critical fraction of the original weights, which together enable good performance. The results show that at the end of the first transmission period (0.91 hours), our approach achieves $86.31 \pm 1.51\%$ accuracy. The next best performance demonstrates $\mathcal{VQ}$ with parameter combination $4/512$. However, it can be seen that the confidence intervals are quite large. Depending on the availability of data, the initial accuracy of $\mathcal{VQ}$ falls in one of two groups: it is either 10%, which corresponds to random classification results, or 85.45%. That is, during some simulation runs, the first transmissions do not suffice to improve accuracy at all. Thus, compared to our approach, $\mathcal{VQ}$ is significantly less stable. An evident solution to overcome this problem would be to reduce the amount of metadata by choosing a smaller codebook size. However, the evaluation for the parameter combinations $4/256$, $4/128$, and $4/64$ clearly demonstrates that the resulting loss in accuracy cannot be compensated by earlier availability of greater amount of original weights.

Finally, we compare our approach to $\mathcal{EG}$ using different group sizes. Evidently, the amount of metadata has a large impact on the performance. While the evaluation in Section V-A demonstrates that better accuracy can be achieved when more priority groups are introduced, the simulation results show an opposite trend. Though a considerable improvement can be seen when the number of groups is increased from two to four, choosing a higher number of groups results in severe performance degradation. Independent of the number of groups, our approach significantly outperforms $\mathcal{EG}$.

VI. Conclusion

Modern small satellite missions that employ onboard data processing can benefit greatly from the use of neuronal networks. At the same time, periodical retraining and timely updates of model parameters from Earth are essential to keep the models operational during the entire mission lifetime. Small satellite communication limitations resulting from short and rare transmission windows, power constraints, and small antenna sizes pose significant difficulties for such updates. The design challenges are further exacerbated by the often asymmetric communication bandwidth, which favors high downlink speeds over uplink speeds. In this paper, we presented a model update transmission protocol, which combines two components – model weight prioritization and vector quantization – to allow for highly efficient updates. Model weights are transmitted incrementally, so that the partially transmitted model achieves high accuracy levels as early as possible. We compared our approach to existing work using prioritization or vector quantization techniques individually. Our results demonstrate that the proposed transmission protocol quickly achieves near optimal results for a number of use cases and considerably outperforms the existing mechanisms.

References

[1] *CubeSat design specification*, The CubeSat Program, Cal Poly SLO, 2022, rev. 14. [Online]. Available: https://www.cubesat.org/cubesatinfo

[2] Y. Cai, K. Guan, E. Nafziger, G. Chowdhary, B. Peng, Z. Jin, S. Wang, and S. Wang, "Detecting in-season crop nitrogen stress of corn for field trials using uav- and cubesat-based multispectral sensing," *IEEE J. Sel. Topics in Applied Earth Obs. and Remote Sens.*, vol. 12, no. 12, 2019.

[3] S. P. Love et al., "NACHOS, a CubeSat-based high-resolution UV-Visible hyperspectral imager for remote sensing of trace gases: system overview and science objectives," in *CubeSats and SmallSats for Remote Sensing V*, vol. 11832, Int. Soc. for Optics and Photonics. SPIE, 2021.

[4] K. Devaraj, R. Kingsbury, M. Ligon, J. Breu, V. Vittaldev, B. Klofas, P. Yeon, and K. Colton, "Dove high speed downlink system," in *Small Satellite Conf.*, 2017.

[5] K. Devaraj, M. Ligon, E. Blossom, J. Breu, B. Klofas, K. Colton, and R. Kingsbury, "Planet high speed radio: Crossing Gbps from a 3U CubeSat," in *Small Satellite Conf.*, 2019.

[6] D. Vasisht, J. Shenoy, and R. Chandra, "L2D2: Low latency distributed downlink for LEO satellites," in *ACM SIGCOMM Conf.*, 2021.

[7] G. Furano, A. Tavoularis, and M. Rovatti, "AI in space: Applications examples and challenges," in *DFT*, 2020.

[8] V. Kothari, E. Liberis, and N. D. Lane, "The final frontier: Deep learning in space," in *HotMobile*, 2020.

[9] A. P. Arechiga, A. J. Michaels, and J. T. Black, "Onboard image processing for small satellites," in *NAECON*, 2018.

[10] V. Leon, G. Lentaris, E. Petrongonas, D. Soudris, G. Furano, A. Tavoularis, and D. Moloney, "Improving performance-power-programmability in space avionics with edge devices: VBN on Myriad2 SoC," *ACM Trans. Embed. Comput. Syst.*, vol. 20, no. 3, mar 2021.

[11] S. K. Johnson, D. Chelmins, D. Mortensen, M. J. Shalkhauser, and R. Reinhart, "Lessons learned in the first year operating software defined radios in space," in *AIAA SPACE 2014 Conf. and Exposition.*

[12] G. Giuffrida, L. Fanucci, G. Meoni, M. Batič, L. Buckley, A. Dunne, C. van Dijk, M. Esposito, J. Hefele, N. Vercruyssen, G. Furano, M. Pastena, and J. Aschbacher, "The Φ-sat-1 mission: The first on-board deep neural network demonstrator for satellite Earth observation," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022.

[13] O. Kondrateva, S. Dietzel, M. Schambach, J. S. Otterbach, and B. Scheuermann, "Filling the gap: Fault-tolerant updates of on-satellite neural networks using vector quantization," in *IFIP Networking Conference*. IEEE, 2023.

[14] O. Kondrateva, S. Dietzel, A. Lößer, and B. Scheuermann, "Parameter prioritization for efficient transmission of neural networks in small satellite applications," in *MedComNet 2023*. IEEE, 2023.

[15] G. Furano, G. Meoni, A. Dunne, D. Moloney, V. Ferlet-Cavrois, A. Tavoularis, J. Byrne, L. Buckley, M. Psarakis, K.-O. Voss, and L. Fanucci, "Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities," *IEEE Aerospace and Electronic Systems Magazine*, vol. 35, no. 12, 2020.

[16] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *NeurIPS*, 2015.

[17] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *ICCV*, 2017.

[18] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, 1948.

[19] G. Zhang, C. Wang, B. Xu, and R. B. Grosse, "Three mechanisms of weight decay regularization," in *ICLR*, 2019.

[20] Y. Gong et al., "Compressing deep convolutional networks using vector quantization," *arXiv:1412.6115*, 2014.

[21] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *CVPR*, 2016.

[22] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," *NeurIPS*, 2014.

[23] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned CP-decomposition," in *ICLR*, 2015.

[24] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NeurIPS Deep Learning and Representation Learning Workshop*, 2015.

[25] J. Yim et al., "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *CVPR*, 2017.

[26] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *NIPS*, 2015.

[27] N. Lee, T. Ajanthan, and P. H. S. Torr, "Snip: single-shot network pruning based on connection sensitivity," in *ICLR*, 2019.

[28] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *ICLR*, 2017.

[29] A. Renda, J. Frankle, and M. Carbin, "Comparing rewinding and fine-tuning in neural network pruning," in *ICLR*, 2020.

[30] D. Wang, A. Mazumdar, and G. W. Wornell, "Compression in the space of permutations," *IEEE Trans. Inf. Theory*, vol. 61, no. 12, 2015.

[31] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "AirNet: Neural network transmission over the air," in *Int. Symp. on Inf. Theory (ISIT)*, 2022.

[32] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, *Object recognition with gradient-based learning*. Springer, 1999.

[33] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Proc. Mag.*, vol. 29, no. 6, 2012.

[34] K. Makantasis, K. Karantzalos, A. Doulamis, and N. Doulamis, "Deep supervised learning for hyperspectral data classification through convolutional neural networks," in *IGARSS*, 2015.

[35] S. Zagoruyko, "92.45% on CIFAR-10 in torch," http://torch.ch/blog/2015/07/30/cifar.html, 2015.

[36] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 dataset," https://www.cs.toronto.edu/ kriz/cifar.html, 2009.

[37] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe, "220 band AVIRIS hyperspectral image data set," 2015. [Online]. Available: https://purr.purdue.edu/publications/1947/1

[38] A. Freimann, M. Dierkes, T. Petermann, C. Liman, F. Kempf, and K. Schilling, "ESTNeT: A discrete event simulator for space-terrestrial networks," *CEAS Space Journal*, vol. 13, 2021.

[39] A. Virdis and M. Kirsche, *Recent Advances in Network Simulation*. Springer, 2019.