 ISO/IEC JTC 1/SC 29/AG 03 N139

**ISO/IEC JTC 1/SC 29/AG 03  
MPEG Liaison and Communication   
Convenorship: KR**

**Document type:** Output Document

**Title:** White Paper on Neural Network Coding

**Status:** Approved

**Date of document:** 2024-01-26

**Source:** ISO/IEC JTC 1/SC 29/AG 03

**Expected action:** None

**Action due date:** None

**No. of pages:** 7 (with cover page)

**Email of Convenor:** kyuheonkim@khu.ac.kr

**Committee URL:** https://isotc.iso.org/livelink/livelink/open/jtc1sc29ag3

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/AG 03 MPEG VIDEO CODING**

**ISO/IEC JTC 1/SC 29/AG 03 N139**

**January 2024, Online**

|  |  |
| --- | --- |
| **Title** | **White Paper on Neural Network Coding** |
| **Source** | **AG 03, MPEG Liaison and communication** |
| **Status** | **Approved** |
| **Serial Number** | **23564** |

**Neural Network Coding (NNC) –**

**Efficient Storage and Inference of Neural Networks for Multimedia Applications**

Artificial neural networks have been adopted for a broad range of tasks in almost every technical field, such as medical applications, transportation, network optimization, big data analysis, surveillance, speech, audio, image and video classification, image and video compression, and many more. Their recent success is based on the feasibility of processing much larger and complex neural networks (deep neural networks, DNNs) than in the past, and the availability of large-scale training data sets. An additional factor for the exponential growth is the appearance of new use cases, such as federated learning with continuous communication between many devices. To effectively reduce bandwidth usage in the communications and reduce the size of networks for inference, achieving an optimal compression ratio must be prioritized. Thus a standard for neural network coding (NNC) has been defined in ISO/IEC 15938-17 “Compression of Neural Networks for Multimedia Description and Analysis”, with the second edition adding new compression tools and support for coding incremental updates of neural networks. This also includes high-level syntax for efficient transportation mechanisms for distributed/federated multi-client scenarios.

Examples of specific applications targeted by the standard include (image) classification, image/video compression or federated training. In many (image) classification applications, trained models (and possibly updates) need to be deployed to a large number of target devices, such as mobile phones or smart cameras. In image/video compression, models adjusted to specific content characteristics may need to be sent frequently to the decoder. In federated training scenarios, updated models need to be exchanged frequently between nodes. These use cases benefit from smaller serialized representations of trained models, and many of the target use cases also benefit from more energy efficient inference with models of reduced complexity.

Incremental coding, one of the main extensions in the second edition, targets neural network updates as a difference signal between a base neural network (i.e. an instance of a trained neural network for the particular use case) and an updated neural network. The updated neural network is typically the result of one of the following operations (this list is considered non-exhaustive), for example:

* The base neural network is retrained with other data or hyper-parameters.
* The base neural network and the updated neural network are compressed versions of the same network with different compression ratio.
* The updated neural network is the result of applying transfer learning, starting from the base neural network.
* The updated neural network uses the base neural network in its structure (possibly retrained end-to-end).

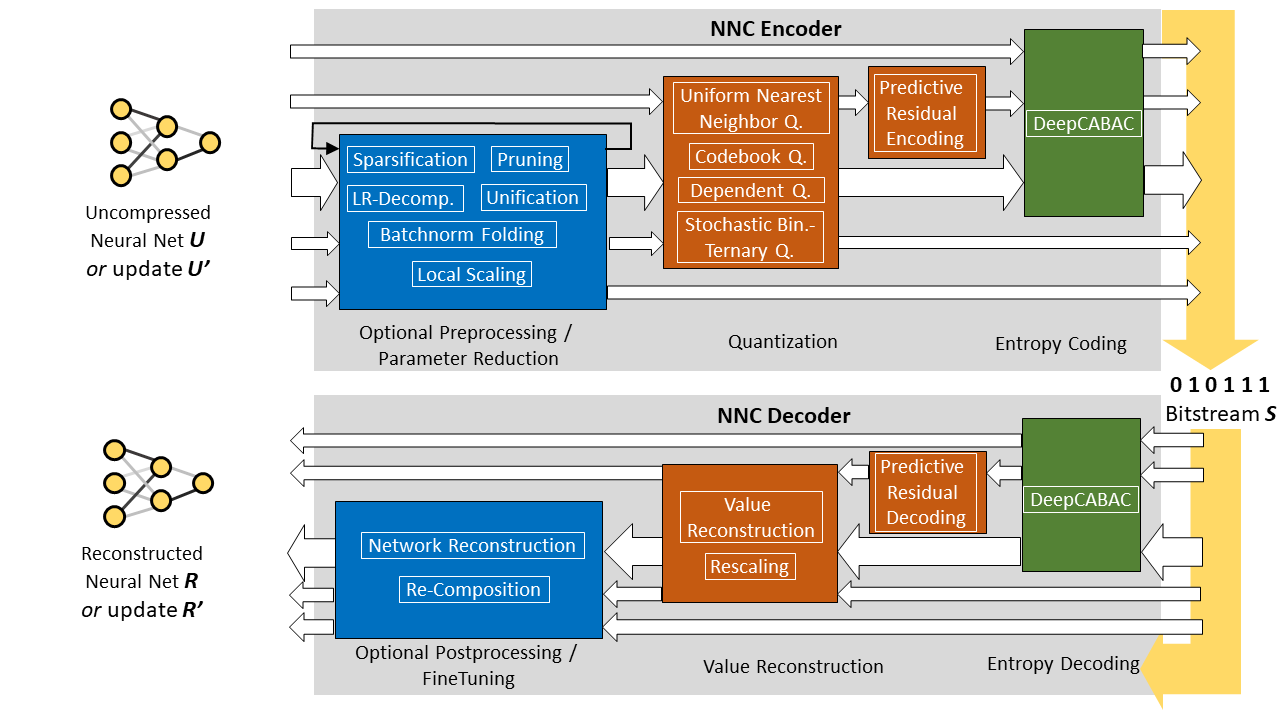
# Scope of the specification and relation to exchange formats

As the coding of neural networks is likely to have a hardware dependent and hardware independent component, the NNC standard is designed as a toolbox of compression technologies. Some of these technologies require specific representations in an exchange format (i.e., sparse representations, adaptive quantization), and thus a normative specification for representing outputs of these technologies is defined. For other technologies (e.g. pruning), that do not require a certain representation, NNC also specifies generic metadata. Therefore, NNC is independent of a particular neural network exchange format. However, interoperability solutions with the most common formats are described in the annexes of the specification, including PyTorch[[1]](#footnote-1), TensorFlow[[2]](#footnote-2), Open Neural Network Exchange (ONNX)[[3]](#footnote-3), and the Neural Network Exchange Format (NNEF)[[4]](#footnote-4). Interoperability with these formats is specified in two ways: either, NNC is used independently by compressing all parameter tensors of a neural network and including the respective NN structure or connection graph into the bitstream, or NNC is used within an external framework by also coding neural network parameters tensor-wise, while all structure data is handled by the external framework.

# Features of the standard

Achieving compact representations of trained neural networks addresses two main goals: (i) providing efficiency when the NN is stored or transmitted and (ii) allowing for resource-efficient inference. The importance of these goals depends on the specific use case. For example, for frequent updates in a federated training scenario involving nodes in a cloud infrastructure the efficient transmission is more important, while for infrequent deployments of a trained NN to an embedded device supporting efficient inference is crucial. Methods addressing the second goal also support the first one, and need to output a representation that can be used directly for inference (at least on specific target platforms), while methods addressing the first one will have additional coding steps requiring decoding at the receiving end.

Fig. 1 NNC edition 2 Overview.



NNC contains a set of processing steps with associated coding tools. Fig. 1 provides an overview of the coding pipelines that can be created. The input to the process is either a regular trained neural network, which can be a stand alone or a base neural network, or an updated network, based on a previous base neural network. The NNC processing steps include the following:

**Parameter reduction methods** process a model to obtain a more compact representation. Examples of such methods include *parameter sparsification, parameter pruning, weight unification,* and *decomposition methods.*

*Sparsification* processes parameters or groups of parameters to produce a sparse representation of the model, e.g., by replacing some weight values with zeros. The sparsification may generate additional metadata (e.g. masks). The sparsification can be structured or unstructured. NNC includes methods for unstructured sparsification with compressibility loss, structured sparsification using micro-structured sparsification. *Unification* processes the parameters to produce groups of similar parameters. Unification does not eliminate or constrain the weights to be zero, but it lowers the entropy of model parameters by making them similar to each other. NNC includes a method for weight unification.

*Pruning* reduces the number of parameters by eliminating parameters or groups of parameters. The procedure results in a dense representation which has less parameters in comparison to the uncompressed model, e.g., by removing some redundant convolution filters from the layers. NNC includes a method for combined pruning and sparsification.

*Decomposition* performs a matrix decomposition operation to change the structure of the weights of a model. This standard includes a method for low rank/low displacement rank for convolutional and fully connected layers.

Along with the reduction methods mentioned above, this standard includes decomposition methods that are introduced and tested as part of a parameter quantization technique. Examples of such methods are batchnorm folding and local scaling adaptation. The parameter reduction methods could be combined or applied in sequence to produce a compact model.

**Parameter quantization methods** reduce the precision of the representation of parameters. If supported by the inference engine, the quantized representation can be used for higher-efficient inference. The methods include uniform, codebook-based and stochastic binary-ternary quantisation. NNC includes methods for uniform quantization, codebook-based quantization, dependent scalar quantization, and iterative QP optimization.

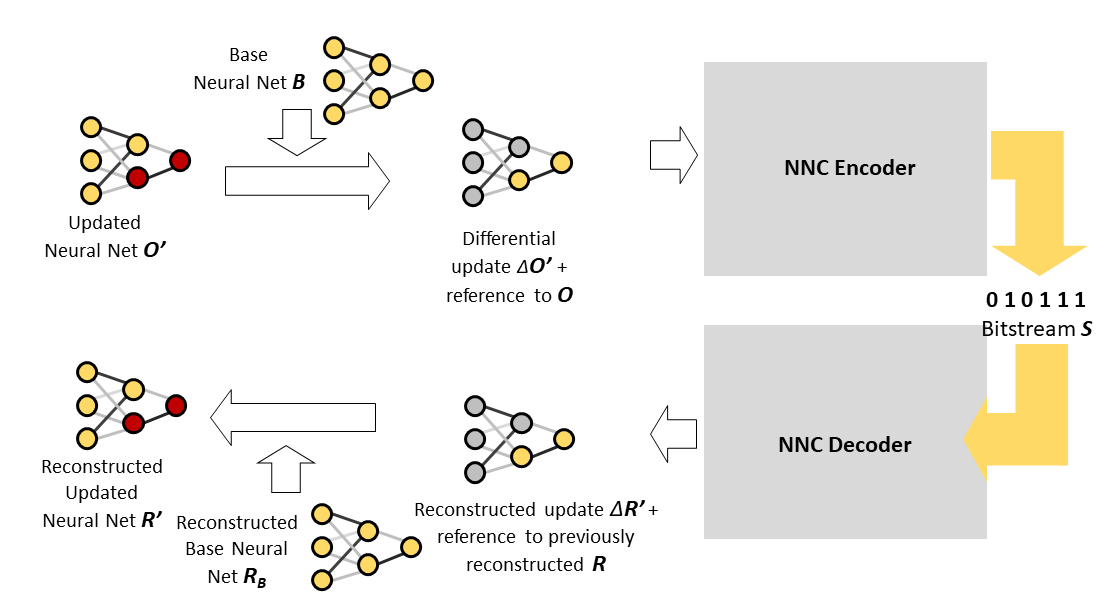
Predictive residual encoding is a specific tool for coding a series of incremental updates (e.g., in federated training scenarios). An update is coded as the difference to the previous update rather than the based model.

**Entropy coding methods** encode the results of parameter quantization methods. This standard includes DeepCABAC as entropy encoding method, which has been extended with temporal context modelling and row skipping to better address the properties of a series of incremental updates.

More details about the methods are described in an overview paper [1].

In the case of incremental coding, an update or differential model is transmitted. Accordingly, a respective NNC-based application needs to generate the differential model before NNC encoding and generate the reconstructed NN from the decoded differential model and a respective base NN. The latter may have been transmitted before or known to the application by other means. This process is shown in Fig. 2.

Fig. 2 NN Reconstruction.



Besides the coding tools, NNC also specifies high-level syntax for efficient carriage of neural networks. With the second edition, high-level syntax was added to also handle multiple and parallel neural network updates in distributed AI scenarios, such as federated learning.

# Performance and reference software

The NNC standard provides a compression efficiency of up to 97% for transparent coding use cases, i.e. without degrading the classification and inference capability of the respective NN. This is reflected by the obtained evaluation results, where compression efficiency in terms of compressed bitrate vs. uncompressed NN bitrate is analyzed. Here, performance metrics for relevant use cases in multimedia for the uncompressed as well as decoded and reconstructed network are evaluated, such as constant top-1 classification accuracy for image classification. In addition, much higher coding gains can be obtained if the classification accuracy is allowed to drop.

The NNC compression performance is evaluated on different verification datasets for use cases of base model and differential model update compression: The base model compression use cases include three models (VGG16, ResNet50, MobileNetV2) for image classification, one model (DCase) for audio classification and an image autoencoder (UC12B). The use cases for differential model update compression include three models (ResNet56, ResNet50, MobileNetV2) for image classification in federated learning (FL) and two models (ResNet18, VIT-B/16) for image classification in transfer federated learning (Transfer-FL). In the latter use case, the client models are already pre-trained on a related dataset.

Table 1: NNC Transparent Coding Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Base Model Compression | | | | |
| Model | ***cr* in %** | **Top-1**  **acc. reconstr.** | **Top-1**  **acc. original** | **Orig. size (bytes)** |
| VGG16 (ImageNet-1k) | 2.98 | 70.51 | 70.93 | 553.43 M |
| ResNet50 (ImageNet-1k) | 6.54 | 74.42 | 74.98 | 102.55 M |
| MobileNetV2 (ImageNet-1k) | 12.18 | 71.13 | 71.47 | 14.16 M |
| DCase (DCASE2019) | 4.12 | 58.15 | 58.27 | 467.26 k |
| Model | ***cr* in %** | **PSNR / SSIM**  **reconstructed** | **PSNR/ SSIM**  **original** | **Orig. size (bytes)** |
| auto-encoder for image compression (CIFAR-100) | 17.34 | 29.98 / 0.954 | 30.13 / 0.956 | 304.72 k |
| Differential Model Update Compression | | | | |
| FL Model  (16 clients) | ***cr* in %** | **Top-1**  **acc. reconstr.**  (≤ 120 rounds) | **Top-1**  **acc. original**  (≤ 100 rounds) | **Orig. size (bytes)** |
| ResNet56 (CIFAR-100) | 2.87 | 57.49  (103 rounds) | 57.79  (99 rounds) | 7.86 G |
| ResNet50 (ImageNet-200) | 2.08 | 52.58  (89 rounds) | 53.00  (75 rounds) | 229.61 G |
| MobileNetV2 (ImageNet-200) | 3.07 | 45.33  (94 rounds) | 45.44  (97 rounds) | 30.79 G |
| Transfer-FL Model  (16 clients) | ***cr* in %** | **Top-1**  **acc. reconstr.**  (≤ 60 rounds) | **Top-1**  **acc. original**  (≤ 50 rounds) | **Orig. size (bytes)** |
| ResNet18  (ImageNet-1k to Pascal VOC) | 0.64 | 72.13 (46 rounds) | 72.61  (49 rounds) | 70.16 G |
| ViT-B/16  (ImageNet-1k to Pascal VOC) | 0.11 | 77.98  (46 rounds) | 78.54  (20 rounds) | 219.68 G |

The coding results for the NN test set are provided in Table 1, showing transparent results at working points with similar accuracy of uncompressed and reconstructed pre-trained NNs for Top‑1 classification accuracy and Peak Signal-to-Noise Ratio (PSNR) / Structural Similarity Index Measure (SSIM) for image compression. For the differential update compression, the number of communication rounds is also shown. The uncompressed use cases were trained with a maximum of 100 and 50 rounds for FL and Transfer-FL respectively. For the use cases with compressed model update communication, the respective number of rounds is reported in which at least 99% of the original Top-1 accuracy (i.e. of the uncompressed scenario) is achieved (with a maximum of 20 additional rounds for compressed FL and 10 additional rounds for compressed Transfer-FL).

The NNC compression capability is given as compression ratio *cr* in Table 1: As an example for VGG16 with *cr* = 2.98%, NNC is able to compress the neural network from ~553 Mbyte to a bitstream size of ~16.5 Mbyte without performance degradation. More details about the evaluation procedure and further results can be found in [1] and [2].

In addition, the standard has been validated in a range of applications, including natural language processing, image restoration and super-resolution, recommenders, implicit neural scene representations, and large language models, achieving compression to 10-20% of the original size without performance loss. Details on the application of NNC to various use cases with neural networks having different layer types can be found in [3].

In a distributed training scenario a model update after a training iteration can be represented at 1% or less of the base model size on average, without sacrificing the classification performance of the neural network.

A complete implementation of the standard is provided as reference software in ISO/IEC 15938-18. In addition, an open source implementation named NNCodec [4] is available.

# References

[1] H. Kirchhoffer et al., "Overview of the Neural Network Compression and Representation (NNR) Standard," in IEEE Transactions on Circuits and Systems for Video Technology, doi: 10.1109/TCSVT.2021.3095970.

[2] D. Becking et al., “Neural Network Coding of Difference Updates for Efficient Distributed Learning Communication,” in IEEE Transactions on Multimedia, doi: 10.1109/TMM.2024.3357198.

[3] ISO/IEC JTC1 SC29 WG04 N0450, "Application and Verification of NNC in Different Use Cases," Jan. 2024.

[4] D. Becking et al., "NNCodec: An Open Source Software Implementation of the Neural Network Coding ISO/IEC Standard," ICML Workshop Neural Compression 2023. https://github.com/fraunhoferhhi/nncodec.

1. https://pytorch.org/ [↑](#footnote-ref-1)
2. https://www.tensorflow.org/ [↑](#footnote-ref-2)
3. https://onnx.ai/ [↑](#footnote-ref-3)
4. https://www.khronos.org/nnef [↑](#footnote-ref-4)