Efficient Federated Learning Tiny Language Models for Cellular Feature Prediction

Daniel Becking

daniel.becking@hhi.fraunhofer.de

AI for Good Global Summit

ITU

Fraunhofer HHI

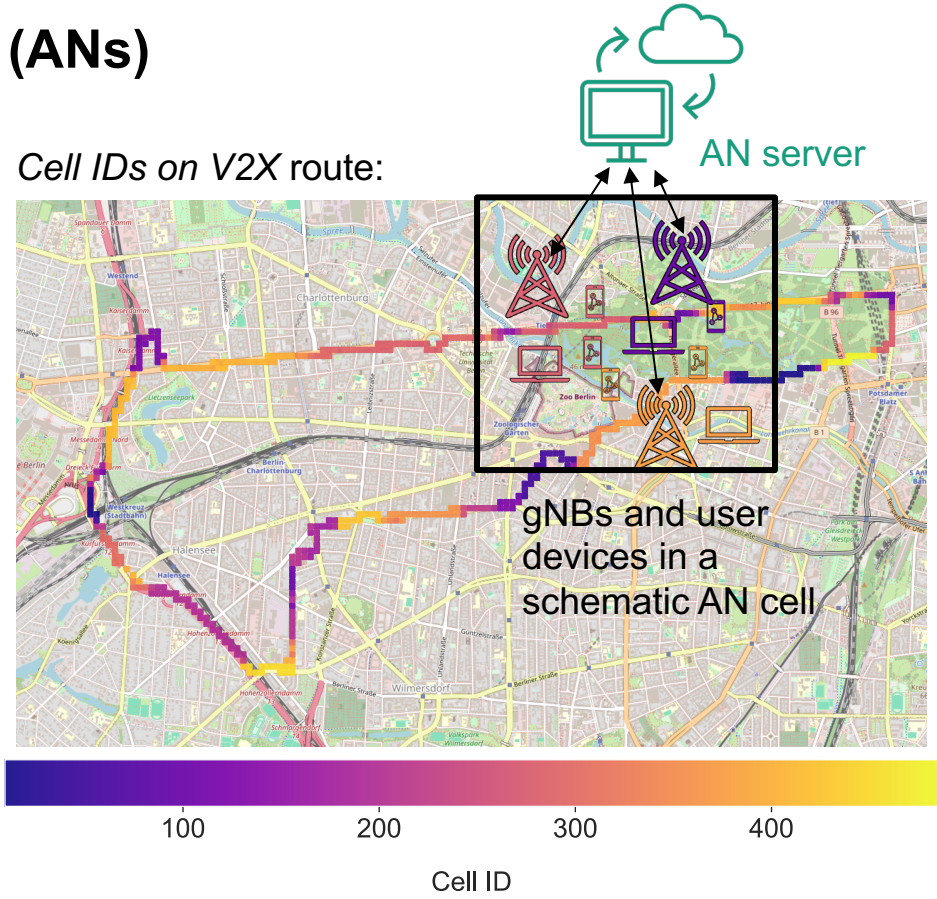
Presentation Roadmap

- **Motivation**: Autonomous Networks & Communication Bottlenecks
- **Methods**: Tiny Language Models, Federated Learning, NNCodec
- **Data**: Berlin V2X Dataset & its Feature Space
- **Experimental Results**: Compression Gains & Prediction Quality
- **Future Work & Conclusion**

daniel.becking@hhi.fraunhofer.de

Motivation: Autonomous Networks (ANs)

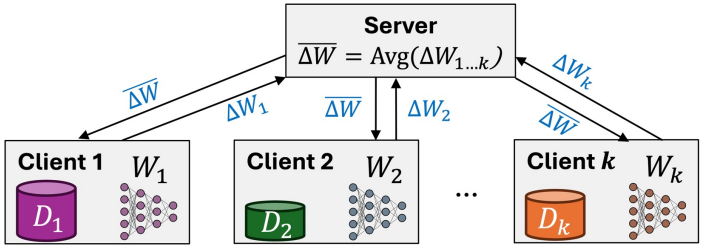
- ANs automatically adjust net configurations based on application requirements and available resources.
- They leverage AI for self-optimization, self-repair, and self-protection.
- Mid-scale self-organizing ANs consist of ~10-100 base stations (gNBs) / cells.
- *Berlin V2X* example: 37 cell IDs observed along a 17 km route through Berlin.
  - We can use Federated Learning to collaboratively predict, model, and optimize cell behavior across this network.



daniel.becking@hhi.fraunhofer.de

Efficient Federated Learning Communication

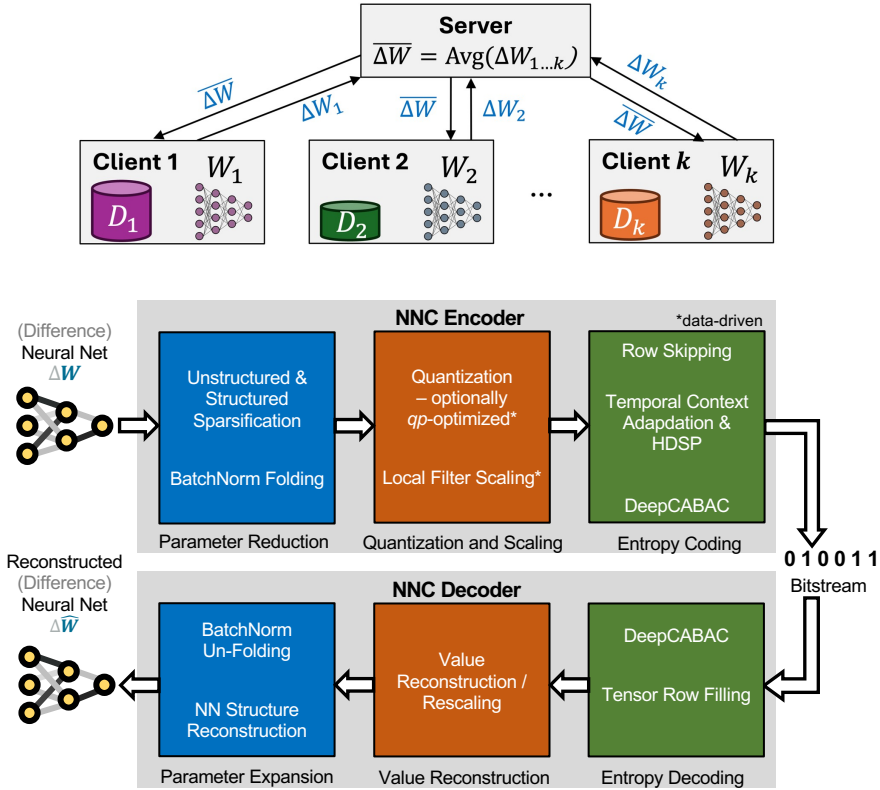
- Federated Learning (FL) allows AN cells to collaboratively train models without sharing raw data, preserving privacy.
- However, frequent exchange of large neural updates introduces **significant communication overhead.**



daniel.becking@hhi.fraunhofer.de

Efficient Federated Learning Communication

- Federated Learning (FL) allows AN cells to collaboratively train models without sharing raw data, preserving privacy.
- However, frequent exchange of large neural updates introduces **significant communication overhead.**
- To reduce this overhead, we apply *NNCodec* [2] – our implementation of the ISO/IEC Neural Network Coding (NNC) standard [3] – to compress model updates exchanged between cells / gNBs and server.

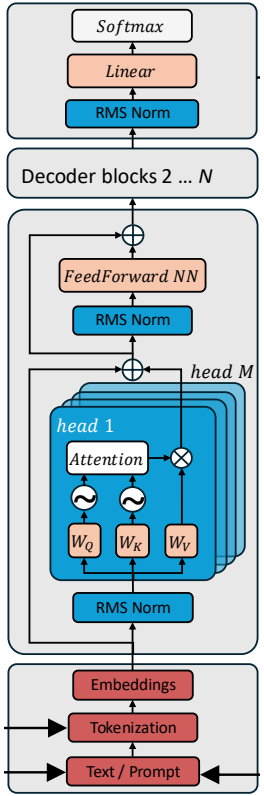


daniel.becking@hhi.fraunhofer.de

Learning Multi-Cellular-Feature-Prediction Using a Single Model

1) Language models excel in capturing complex correlations in data through the attention mechanism, often at a high computational cost.
   → Motivates the use of efficient, **tiny language models** (TLM).
2) Lightweight, specialized tokenizer for cellular features in telecommunication.

1)



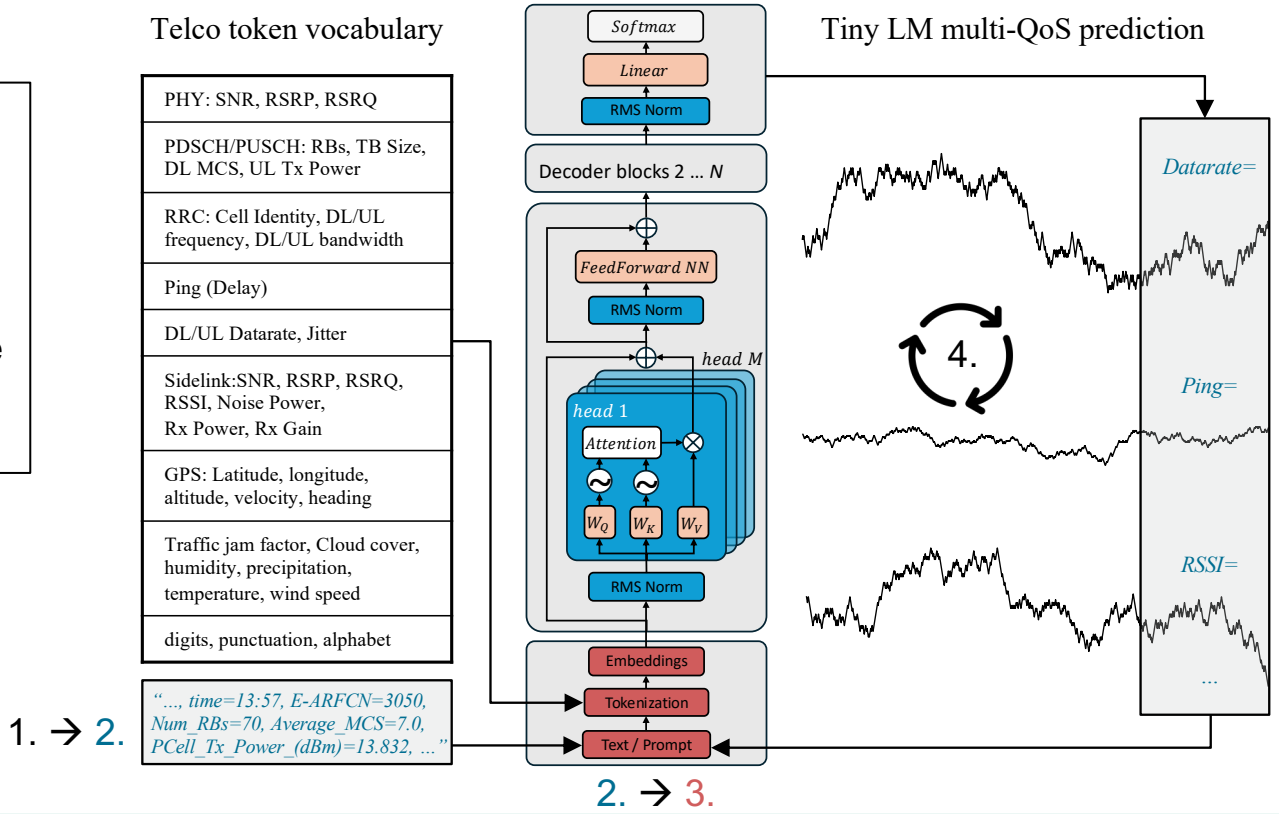
2) Telco token vocabulary

PHY: SNR, RSRP, RSRQ
PDSCH/PUSCH: RBs, TB Size, DL MCS, UL Tx Power
RRC: Cell Identity, DL/UL frequency, DL/UL bandwidth
Ping (Delay)
DL/UL Datarate, Jitter
Sidelink:SNR, RSRP, RSRQ, RSSI, Noise Power, Rx Power, Rx Gain
GPS: Latitude, longitude, altitude, velocity, heading
Traffic jam factor, Cloud cover, humidity, precipitation, temperature, wind speed
digits, punctuation, alphabet

daniel.becking@hhi.fraunhofer.de

Learning Multi Cellular Feature Prediction Using a Single Model

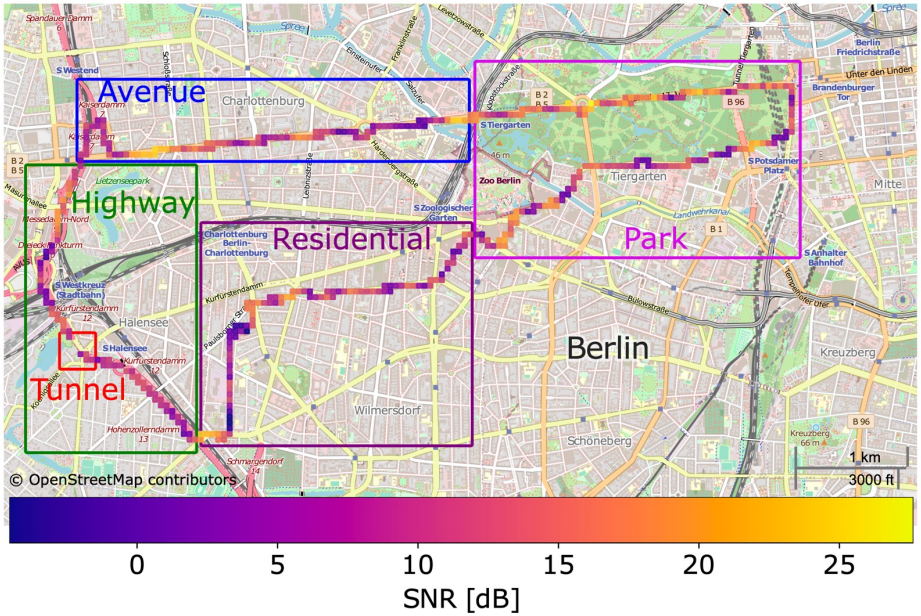
1. Tabular config / feature format
2. Text / strings
3. Tokens
4. Iterative, autoregressive next-token-prediction using TLM



daniel.becking@hhi.fraunhofer.de

Berlin V2X [4]

A ML Dataset from Multiple Vehicles & Radio Access Technologies

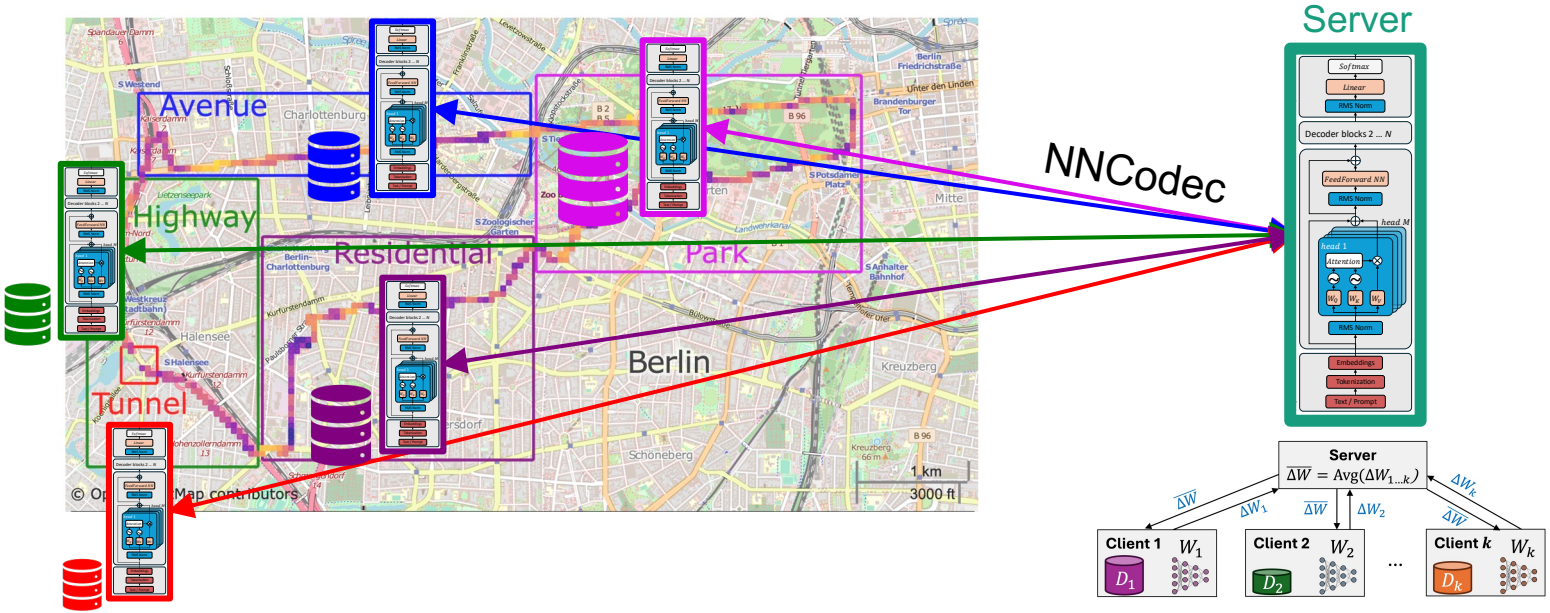


3 days, 4 UEs (cars), 17 rounds, >160 Features

Data category	Source	Tool	Sampling interval	Features
Cellular	DME	Mobile Insight	10 ms	PHY: SNR, RSRP, RSRQ, RSSI
			20 ms	PDSCH/PUSCH: RBs, TB Size, DL MCS, UL Tx Power
			Event-based	RRC: Cell Identity, DL/UL frequency, DL/UL bandwidth
		ping	1 s	Delay
		iperf	1 s	DL Datarate, Jitter
	Server	iperf	1 s	UL Datarate, Jitter
Sidelink	SDR UE	tcpdump	Event-based	SNR, RSRP, RSRQ, RSSI, Noise Power, Rx Power, Rx Gain
Position	GPS		1 s	Latitude, Longitude, Altitude, Velocity, Heading
Side information	Internet database	HERE API	5 min	Traffic Jam Factor, Traffic Street Name, Traffic Distance
		DarkSky	1 hour	Cloud cover, Humidity, Precipitation Intensity & Probability, Temperature, Pressure, Wind Speed

daniel.becking@hhi.fraunhofer.de

Federated Learning Berlin V2X with Tiny Language Models (TLMs)



Each cell equipped with a TLM collecting local data and transmitting model updates to the Fed server.

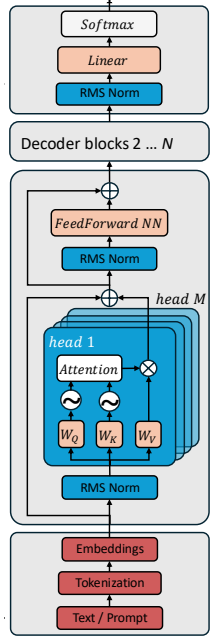
daniel.becking@hhi.fraunhofer.de

Exemplary Input/Output to/of the TLM

Input prompt:

"time = 1 3 5 7 direction = uplink measured_qos = delay device = pc4 target_datarate = 4 0 0 0 0 0 --> PCell_E-ARFCN = 3 0 5 0 . 0 0 0 PCell_Downlink_Num_RBs = 7 0 . 0 0 0 PCell_Downlink_TB_Size = 1 7 1 2 . 0 0 0 PCell_Downlink_RBs_MCS_0 = 3 . 0 0 0 PCell_Downlink_RBs_MCS_1 = 4 . 0 0 0 PCell_Downlink_RBs_MCS_2 = PCell_Downlink_RBs_MCS_3 = PCell_Downlink_RBs_MCS_4 = PCell_Downlink_RBs_MCS_5 = 1 1 . 0 0 0 PCell_Downlink_RBs_MCS_6 = 3 2 . 0 0 0 PCell_Downlink_RBs_MCS_7 = 3 . 0 0 0 PCell_Downlink_RBs_MCS_8 = 1 2 . 0 0 0 PCell_Downlink_RBs_MCS_9 = PCell_Downlink_RBs_MCS_10 = PCell_Downlink_RBs_MCS_11 = PCell_Downlink_RBs_MCS_12 = PCell_Downlink_RBs_MCS_13 = PCell_Downlink_RBs_MCS_14 = PCell_Downlink_RBs_MCS_15 = PCell_Downlink_RBs_MCS_16 = PCell_Downlink_RBs_MCS_17 = PCell_Downlink_RBs_MCS_18 = PCell_Downlink_RBs_MCS_19 = PCell_Downlink_RBs_MCS_20 = PCell_Downlink_RBs_MCS_21 = PCell_Downlink_RBs_MCS_22 = PCell_Downlink_RBs_MCS_23 = PCell_Downlink_RBs_MCS_24 = PCell_Downlink_RBs_MCS_25 = PCell_Downlink_RBs_MCS_26 = PCell_Downlink_RBs_MCS_27 = PCell_Downlink_RBs_MCS_28 = PCell_Downlink_RBs_MCS_29 = 5 . 0 0 0 PCell_Downlink_RBs_MCS_30 = PCell_Downlink_RBs_MCS_31 = PCell_Downlink_Average_MCS = 7 . 0 0 0 PCell_Uplink_Num_RBs = 4 8 3 7 . 0 0 0 PCell_Uplink_TB_Size = 1 0 9 7 8 1 . 0 0 0 PCell_Uplink_Tx_Power_(dBm) = 1 3 . 8 3 2 PCell_Downlink_frequency = 3 0 5 0 . 0 0 0 PCell_Uplink_frequency = 2 1 0 5 0 . 0 0 0 PCell_Downlink_bandwidth_MHz = 2 0 PCell_freq_MHz = 2 6 0 0 . 0 0 0 PCell_Uplink_bandwidth_MHz = 2 0 PCell_Band_Indicator = 7 . 0 0 0 PCell_Cell_ID = 4 3 0 . 0 0 0 PCell_Cell_Identity = 3 3 8 0 2 2 4 7 . 0 0 0 --> Latitude = 5 2 . 5 1 4 Longitude = 1 3 . 3 4 8 Altitude = 3 1 . 2 0 0 speed_kmh = 1 9 . 0 7 6 COG = 8 2 . 7 0 0 Pos_in_Ref_Round = 1 6 1 9 8 . 4 1 7 --> precipIntensity = 0 . 1 3 2 precipProbability = 0 . 0 5 0 temperature = 2 0 . 4 6 0 apparentTemperature = 2 0 . 4 6 0 dewPoint = 1 3 . 1 8 0 humidity = 0 . 6 3 0 pressure = 1 0 1 9 . 0 0 0 windSpeed = 2 . 6 8 0 cloudCover = 0 . 9 6 0 uvIndex = 4 . 0 0 0 visibility = 1 6 . 0 9 3 Traffic_Jam_Factor = 2 . 5 6 7 Traffic_Street_Name = Großer Stern Traffic_Distance = 3 . 4 7 3 --> "

Tiny LM



Token-by-token QoS string generation:

"PCell_RSRP_1 = - 1 0 5 . 0 7 6 PCell_RSRP_2 = - 1 0 7 . 7 0 3 PCell_RSRP_max = - 1 0 5 . 0 7 6 PCell_RSRQ_1 = - 8 . 4 9 8 PCell_RSRQ_2 = - 7 . 5 6 7 PCell_RSRQ_max = - 7 . 4 4 5 PCell_RSSI_1 = - 7 6 . 5 7 9 PCell_RSSI_2 = - 8 0 . 1 3 6 PCell_RSSI_max = - 8 0 . 1 3 6 PCell_SNR_1 = 9 . 9 3 9 PCell_SNR_2 = 1 0 . 8 2 7 jitter = 0 . 0 0 8 ping_ms = 2 6 . 2 4 0 datarate = 3 9 4 0 0 0 . 0 0 0"

daniel.becking@hhi.fraunhofer.de

Main Configuration Features:

- **PCell_E-ARFCN**: Absolute Radio Frequency Channel Number; this is a static frequency configuration.
- **PCell_Down / Uplink_Num_RBs**: Number of down / uplink Resource Blocks allocated, part of network configuration.
- **PCell_Down / Uplink_TB_Size** : Transport Block Size for down / uplink, related to throughput configuration.
- **PCell_Downlink_RBs_MCS_0** to **_31**: Modulation and Coding Scheme (MCS) indices for different Resource Blocks, related to encoding configuration for transmission.
- **PCell_Downlink_Average_MCS**: Average MCS for downlink, tied to encoding and modulation configuration.
- **PCell_Uplink_Tx_Power_(dBm):** Transmission power for uplink, a configuration for signal strength.
- **PCell_Down / Uplink_frequency**: Down / Uplink frequency configuration.
- **PCell_Down / Uplink_bandwidth_MHz**: Down / Uplink bandwidth, part of the cell's setup.
- **PCell_Band_Indicator**: Band indicator, part of the frequency and band configuration.
- **PCell_freq_MHz**: Frequency in MHz, which is a configuration feature.

Quality Features (e.g., for Feedback and Optimization):

- **ping_ms**: Ping time in milliseconds, a direct measure of network latency and quality.
- **datarate**: Data rate or throughput, indicating connection speed and quality.
- **jitter**: Variation in latency, a quality metric reflecting stability.
- **PCell_RSRP_1**, **PCell_RSRP_2**, **PCell_RSRP_max**: Reference Signal Received Power; indicates signal strength quality.
- **PCell_RSRQ_1**, **PCell_RSRQ_2**, **PCell_RSRQ_max**: Reference Signal Received Quality; indicates signal-to-noise ratio and interference.
- **PCell_RSSI_1**, **PCell_RSSI_2**, **PCell_RSSI_max**: Received Signal Strength Indicator; indicates the power of the received signal.
- **PCell_SNR_1**, **PCell_SNR_2**: Signal-to-Noise Ratio, a measure of signal quality.

daniel.becking@hhi.fraunhofer.de

GPS Features:

'Latitude', 'Longitude', 'Altitude', 'speed_kmh', 'COG', 'Pos_in_Ref_Round'

Side Information Features:

'precipIntensity', 'precipProbability', 'temperature', 'apparentTemperature', 'dewPoint', 'humidity', 'pressure', 'windSpeed', 'cloudCover', 'uvIndex', 'visibility', 'Traffic_Jam_Factor', 'Traffic_Street_Name', 'Traffic_Distance'

Modular Design

Default order of features:

In: Time stamp → Main Configuration Features → GPS Features → Side Information Features

Out: Quality Features

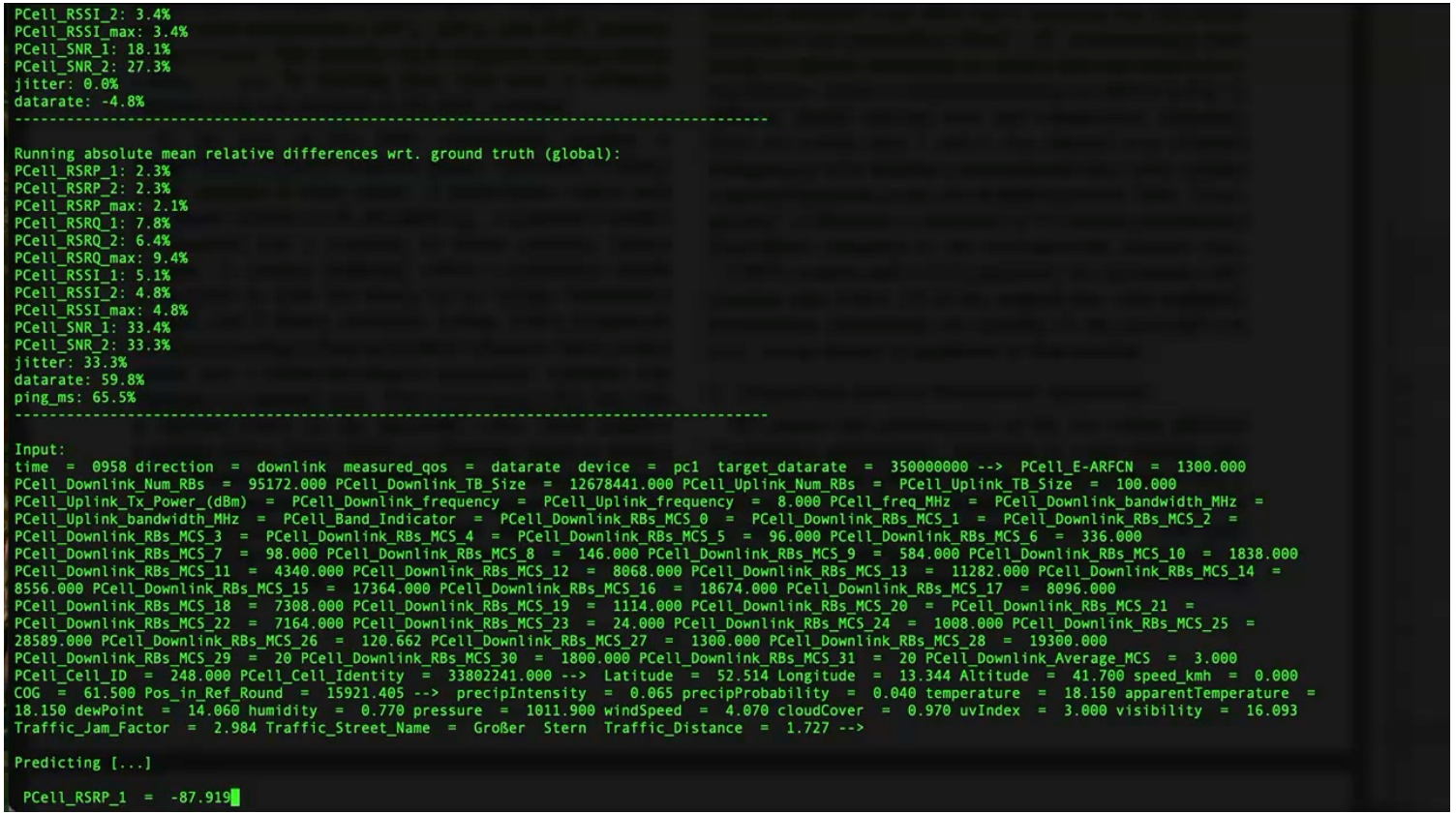
Also works the other way round, depending on use case, for instance:

In: Time stamp → Quality Features → GPS Features → Side Information Features

Out: Main Configuration Features

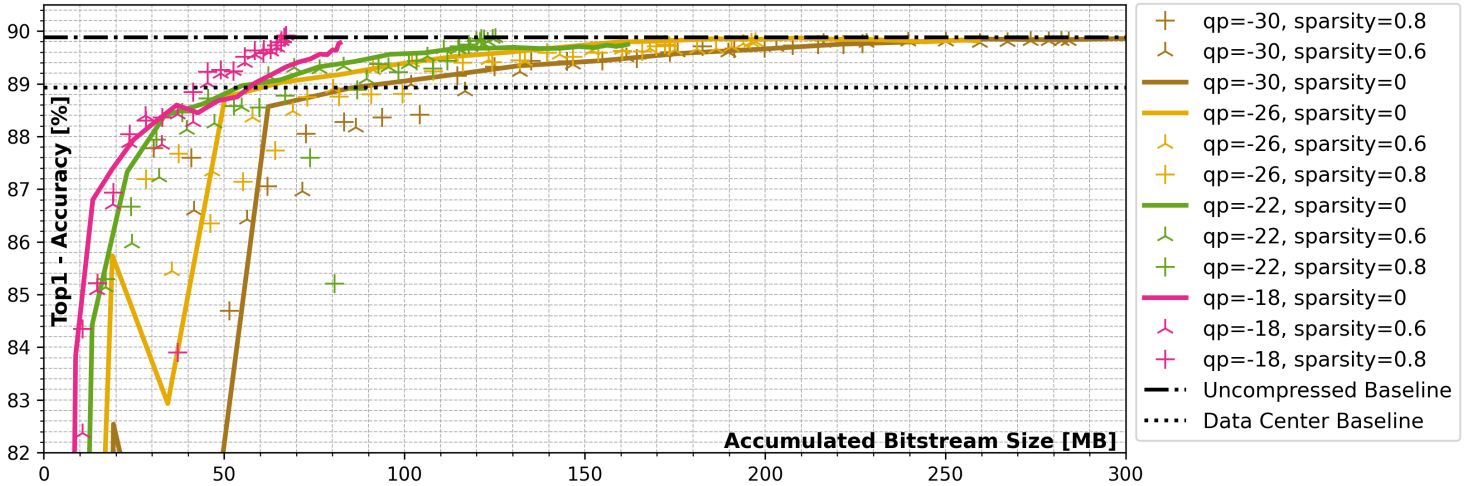
daniel.becking@hhi.fraunhofer.de

QoS Feature Prediction Demo (recorded)



daniel.becking@hhi.fraunhofer.de

Coding Results Using NNCodec



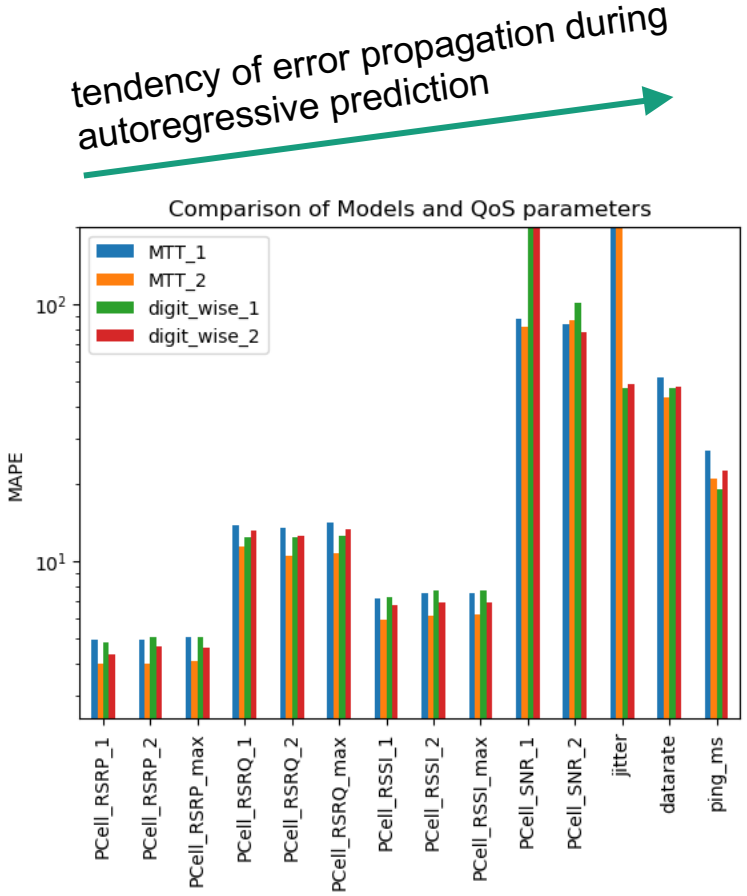
Size	lr	qp	Sparsity	Performance		Data [MB]		Comp.	Perf. Degradation	
				Acc [%]	PPL	Uncomp.	Comp.	Ratio [%]	Acc [%]	PPL
0	3e-4	-26	0%	89.35	1.35	424.64	12.84	3.02	-0.18	+0.01
1	5e-5	-22	0%	89.70	1.35	10,095.84	25.10	0.25	-0.33	+0.00
1	3e-4	-18	80%	89.88	1.34	10,095.84	67.16	0.67	+0.03	-0.00
2	5e-5	-22	60%	89.73	1.35	31,874.56	49.95	0.16	-0.34	+0.00
2	3e-4	-22	0%	89.70	1.35	31,874.56	487.12	1.52	+0.11	-0.00

Communication overhead compressed to below 1% with negligible loss in model performance

daniel.becking@hhi.fraunhofer.de

Prediction Quality

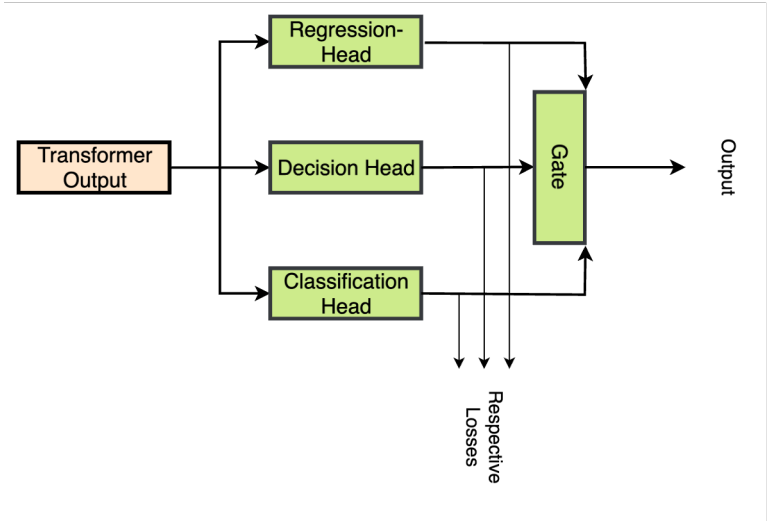
- 2-10% mean relative differences wrt. ground truth for RSRP, RSRQ, and RSSI, 20-40% for data rate and ping
  - SOTA compared to short-horizon forecasts on *Berlin V2X* [6]
- Original data is incomplete (different sampling rates across feature types)
  - Our approach learns to fill in the gaps and generalize on unseen sequences of cellular features



daniel.becking@hhi.fraunhofer.de

Future Work

- Apply approach at different **hierarchy** levels: base station, cell, cell clusters, city areas, cities, …
- "**Mixed-Token Transformers (MTT)"** for both numerical and textual tokens vs. digit-by-digit prediction [currently under review]
- **Domain adaptation** and **personalization** techniques to accommodate heterogeneities and specialized scenarios, such as tunnel environments or events (concerts, sports, …)



Model	Context length	MAPE[a]	# inference[b]	time[c]
Digit-wise, size 1	840	20.16	268	7.25
Digit-wise, size 2	840	18.40	268	9.96
MTT, size 1	436	20.06	82	1.68
MTT, size 2	436	17.80	82	1.87

[a]MAPE excludes outliers *PCell_SNR_1* and *jitter*, see Fig 7.
[b]average number of forward passes to complete a sample with QoS predictions.
[c]average number of seconds for test-set sample completion on a MacBook Pro M3.

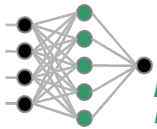
→ *MTT is computationally more efficient and achieves more accurate predictions*

daniel.becking@hhi.fraunhofer.de

Conclusion

- We introduced a new use case for autonomous network cells using Tiny Language Models (TLMs) to efficiently process diverse cellular input features and jointly predict and complement these features.
- The virtual Autonomous Network (AN) cells are integrated into a Federated Learning (FL) setting – training the *Berlin V2X* cellular dataset – enabling collaborative, privacy-preserving, and transferable training across geographic areas.
- Communication efficiency is achieved by compressing incremental updates with *NNCodec* to less than 1% of their original size with negligible performance degradation, cutting gigabytes of communication overhead per training round.

- Code is available at https://github.com/d-becking/nncodec2



NNCodec
NEURAL NETWORK CODING SOFTWARE

daniel.becking@hhi.fraunhofer.de

Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute (HHI)

THANK YOU FOR YOUR ATTENTION.

--

Daniel Becking
Department of Artificial Intelligence
Efficient Deep Learning Group

✉ daniel.becking@hhi.fraunhofer.de
🏠 https://www.hhi.fraunhofer.de/
☎ +49 30 31002-406
🏢 Einsteinufer 37, 10587 Berlin

References

[1] D. Becking et al., “Efficient Federated Learning Tiny Language Models for Mobile Network Feature Prediction,” in Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit): Posters (PST), 2025.

[2] D. Becking et al., “NNCodec: An Open Source Software Implementation of the Neural Network Coding ISO/IEC Standard,” in ICML Neural Compression Workshop, Jul. 2023. https://github.com/d-becking/nncodec2

[3] ISO, “Information technology — Multimedia content description interface - Part 17: Compression of neural networks for multimedia content description and analysis,” International Organization for Standardization (ISO), Standard ISO/IEC 15938-17:2024.

[4] R. Hernangómez et al., "Berlin V2X: A Machine Learning Dataset from Multiple Vehicles and Radio Access Technologies," in IEEE VTC2023-Spring, Florence, Italy, Jun. 2023, pp. 1–5.

[5] D. Becking et al., “Neural Network Coding of Difference Updates for Efficient Distributed Learning Communication,” in IEEE Transactions on Multimedia, vol. 26, pp. 6848–6863, 2024.

[6] B. Denizer and O. Landsiedel, “BandSeer: Bandwidth Prediction for Cellular Networks,” in IEEE 49th Conference on Local Computer Networks (LCN), Oct. 2024, pp. 1-8.

daniel.becking@hhi.fraunhofer.de

Supplementary Material

daniel.becking@hhi.fraunhofer.de

TLM Architecture and Sizes

TABLE I
LLAMA CONFIGURATIONS AND RESULTING MODEL SIZES.

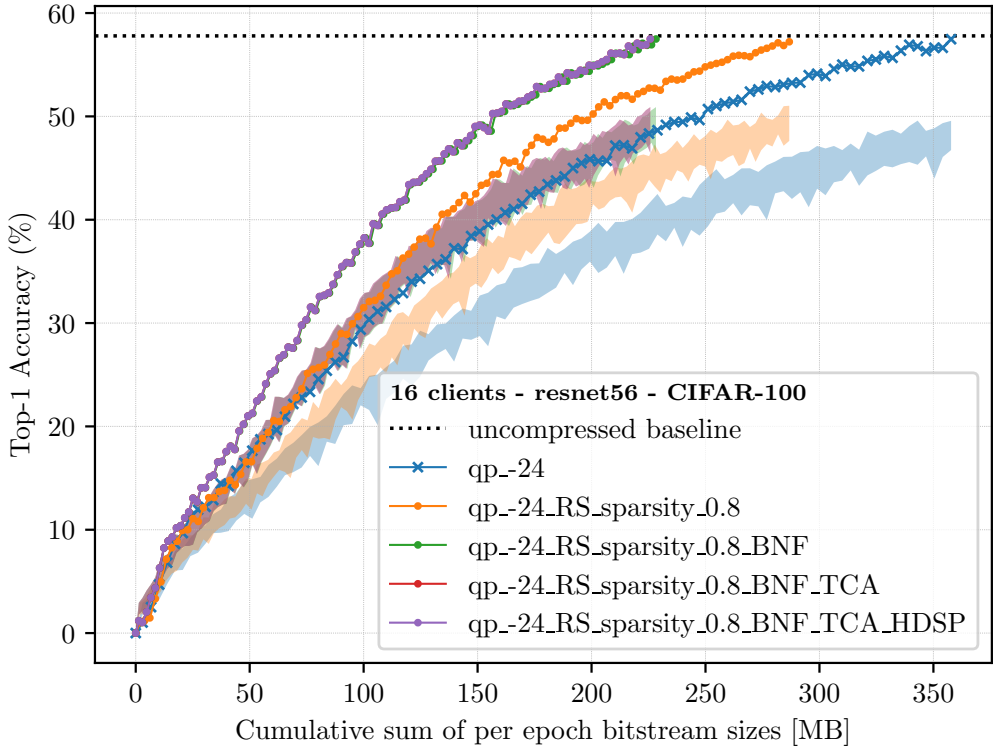
Llama configuration	Size = 1	Size = 2
dim[a]	288	512
n_layers[b]	10	10
n_heads[c]	6	8
n_kv_heads[d]	6	8
vocab_size[e]	237	237
max_seq_len[f]	[536, 840]	[536, 840]
Number of tensors	107	107
Tensor elements	10,429,446	32,926,758
Memory	41.72MB	131.71 MB

a Size of embedding vectors for tokens, i.e., the base size of the hidden representation.
b Number of transformer blocks, each comprising multi-head attention and Feed-forward neural network (FFN) components.
c Number of key-value heads (which can be decoupled from the number of query heads).
d Number of attention heads in the multi-head self-attention mechanism.
e Size of the vocabulary, representing the number of unique tokens the model can handle.
f Maximum input sequence length the model can process.

Our models are based on A. Karpathy's "*llama2.c*" https://github.com/karpathy/llama2.c, 2023, License: MIT; accessed: April 24, 2025.

daniel.becking@hhi.fraunhofer.de

Incremental dNN Coding Gains of Individual (data-free) Tools



1. Uniform quantization
2. + (Un)structured sparsification methods
   + RS (row skipping)
3. + FedBNF: incremental (Federated) BatchNorm Folding
4. + TCA: Temporal Context Adaptation
5. + HDSP: History Dependent Significance Probability

Federated Learning from scratch

(Extended MPEG CTC use case)

daniel.becking@hhi.fraunhofer.de

NNCoding of Other Model Types in Federated and Split Learning

[5]

model	paradigm	#clients	data	uncompressed communication			compressed communication			
				acc. [%]	$\#_{\text{rounds}}$	$\sum$data [GB]	Δacc. [%]	$\Delta\#_{\text{rounds}}$	$\sum$data [MB]	CR (*w/o HLS*) [%]
ResNet-20	FL	16	CIFAR-10	85.59	99	3.42	-0.84	+4	65	1.89 (*1.72*)
ResNet-56	FL	16	CIFAR-100	57.79	100	7.86	-0.30	+18	226	2.87 (*2.61*)
ResNet-50	FL	16	ImageNet-200	53.00	75	229.61	-0.42	+14	4,778	2.08 (*2.07*)
MobileNetV2	FL	16	ImageNet-200	45.44	97	30.79	-0.11	-3	944	3.07 (*3.01*)
EfficientNet-B0	FL	16	ImageNet-200	53.78	90	49.12	-0.04	+1	1,334	2.72 (*2.67*)
ResNet-18	FL^{τ}	16	Pascal VOC	72.61	49	70.16	-0.48	-3	446	0.64 (*0.63*)
ViT-B/16	FL^{τ}	16	Pascal VOC	78.54	20	219.68	-0.56	+26	246	0.11 (*0.11*)
ResNet-20	SL	20	CIFAR-10	82.00	99	655.36	-0.54	-9	18,049	2.75
ResNet-56	SL	20	CIFAR-100	48.14	99	2,621.44	-0.30	-2	88,570	3.38
MobileNetV2	SL	20	ImageNet-200	43.86	96	6,021.06	-0.17	-16	261,328	4.34
EfficientNet-B0	SL	20	ImageNet-200	52.28	69	11,239.31	-0.74	0	454,019	4.04

	CIFAR-10, ResNet-20			CIFAR-100, ResNet-56			ImageNet-200, MobileNetV2			Pascal VOC, ResNet-18		
	Δacc. [%]	$\#_{\text{rounds}}$	$\sum$data [MB]	Δacc. [%]	$\#_{\text{rounds}}$	$\sum$data [MB]	Δacc. [%]	$\#_{\text{rounds}}$	$\sum$data [MB]	Δacc. [%]	$\#_{\text{rounds}}$	$\sum$data [MB]
STC [27]	-1.59 ×	116	184.06	-3.20 ×	119	604.94	-0.45 ✓	117	2,329.87	-0.48 ✓	57	1,137.53
FedZip [30]	-0.67 ✓	105	88.81	-0.34 ✓	111	456.72	-2.04 ×	112	1,891.23	-0.69 ✓	53	1,197.86
dNNC	-0.84 ✓	103	**58.90**	-0.30 ✓	118	**205.64**	-0.11 ✓	94	**927.88**	-0.48 ✓	46	**443.29**

✓means the training converged within the given number of communication rounds t, × means the opposite

- Tested in a wide range of use cases and configurations
- Superior to state-of-the-art with compression ratios below 1%
- Reduces energy consumption by up to 94% [5]

daniel.becking@hhi.fraunhofer.de

Code usage: `from nncodec.fl import NNClient, NNCFedAvg`

- The `nnc_fl.py` file implements a base script for communication-efficient Federated Learning with NNCodec. It imports the `NNClient` and `NNCFedAvg` classes — specialized NNC-Flower objects — that are responsible for establishing and handling the compressed FL environment.
- To start NNCoded efficient FL with TLMs on the Berlin V2X dataset, execute:

```
python example/nnc_fl.py --dataset=V2X --dataset_path=<your_path>/v2x --model=tinyllama
--model_rand_int --num_clients=5 --epochs=30 --compress_upstream --compress_downstream
--err_accumulation --compress_differences --qp=-18 --batch_size=8 --max_batches=300
--max_batches_test=150 --sparsity=0.8 --struct_spars_factor=0.9 --TLM_size=1 --tca
--tokenizer_path=./example/tokenizer/telko_tokenizer.model
```

The pre-tokenized Berlin V2X dataset can be downloaded here:
https://datacloud.hhi.fraunhofer.de/s/CcAeHRoWRqe5PiQ

- Evaluation can be started by executing:

```
python example/eval.py --model_path=<your_path>/best_tinyllama_.pt --batch_size=1
--dataset=V2X --dataset_path=<your_path>/v2x --model=tinyllama --TLM_size=1
--tokenizer_path=./example/tokenizer/telko_tokenizer.model
```


daniel.becking@hhi.fraunhofer.de

--qp 'Quantization parameter (QP) for NNs (default: -32)'

--nonweight_qp 'QP for non-weights, e.g., 1D or BatchNorm params (default: -75)'

--opt_qp 'Enables layer-wise QP modification based on relative layer size within NN'

--use_dq 'Enables dependent scalar / Trellis-coded quantization'

--bitdepth 'Optional: integer-aligned bitdepth for limited precision [1, 31] bit; note: overwrites QPs.'

--bnf 'Enables incremental BatchNorm Folding (BNF)'

--sparsity 'Introduces mean- & std-based unstructured sparsity [0.0, 1.0] (default: 0.0)'

--struct_spars_factor 'Introduces structured per-channel sparsity (based on channel means); requires sparsity > 0 (default: 0.9)'

--row_skipping 'Enables skipping tensor rows from arithmetic coding that are entirely zero'

--tca 'Enables Temporal Context Adaptation (TCA)'

--compress_differences 'Weight differences wrt. to base model (dNN) are compressed, otherwise full base models (NN) are communicated'

--model_rand_int 'If set, model is randomly initialized, i.e., w/o loading pre-trained weights'

--num_clients 'Number of clients in FL scenario (default: 2)'

--compress_upstream 'Compression of clients-to-server communication'

--compress_downstream 'Compression of server-to-clients communication'

--err_accumulation 'If set, quantization errors are locally accumulated ("residuals") and added to NN update prior to compression'

daniel.becking@hhi.fraunhofer.de