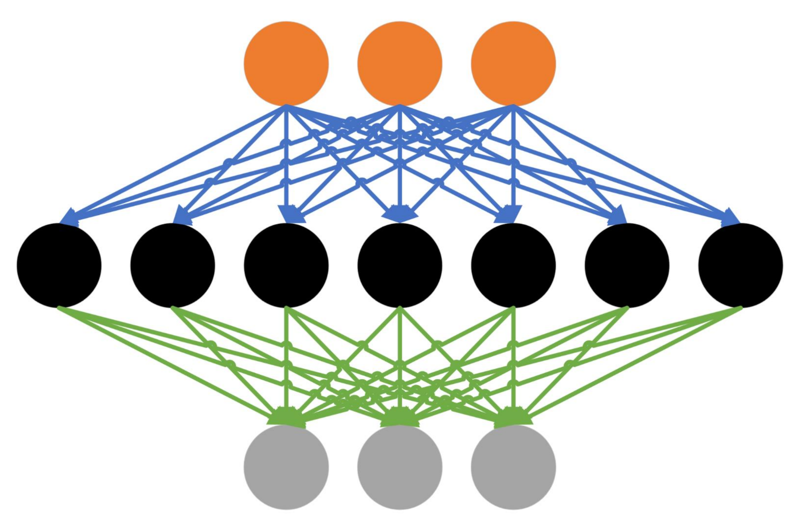
SELF-COMPRESSED NANO GPT

Imagination

József Konczer *jozsef.konczer@imgtec.com* **Timothy Gale** *timothy.gale@imgtec.com* **James Imber** *james.imber@imgtec.com*

MOTIVATION

Consider a linear autoencoder which maps 3 input dimensions to N latent dimensions and back:



$$\underset{A,B}{\operatorname{argmin}} \sum_i (x_i - ABx_i)^2$$

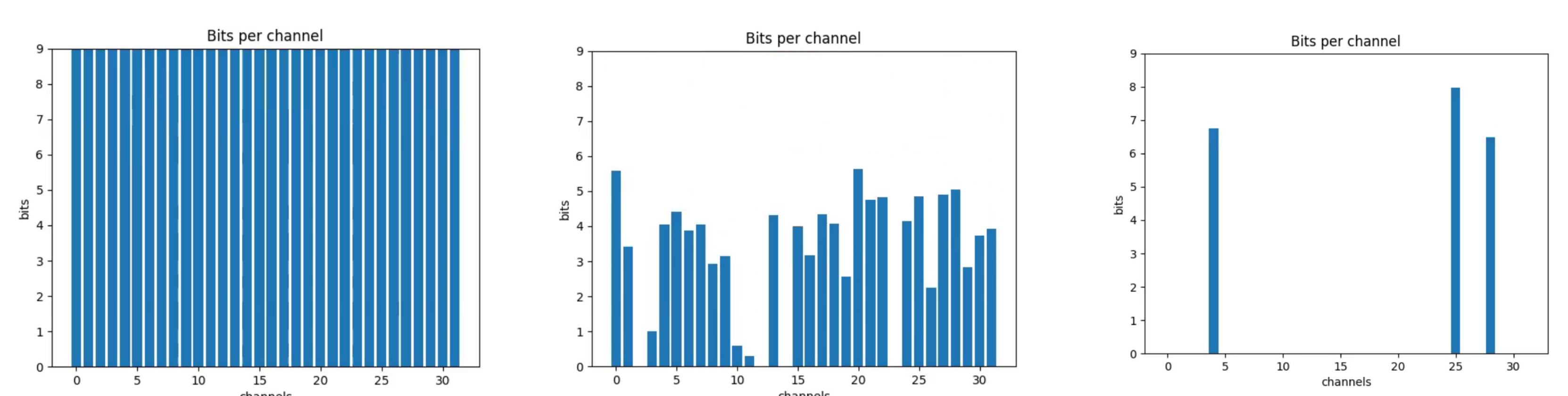
$$y = ABx$$

Linear autoencoder

We used a general method to determine the optimal number of bits and to encode weights or activations using gradient descent. For this we used a quantization that is differentiable in the number of bits [1,2]:

$$q(x,b,e) = 2^e \lfloor \min(\max(2^{-e}x, -2^{b-1}), 2^{b-1}-1) \rceil$$

$$\underset{A,B,b,e}{\operatorname{argmin}} \sum_i \left(\left(x_i - Aq(Bx_i, \boldsymbol{b}, \boldsymbol{e})\right)^2 + \gamma \sum_j b_j \right)$$



Self-compressing training of a linear autoencoder with 3 input channels

RESULTS

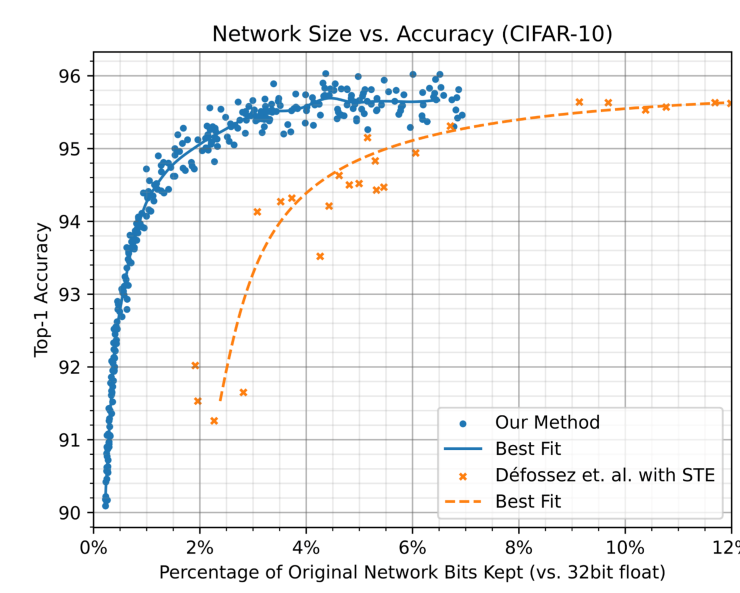
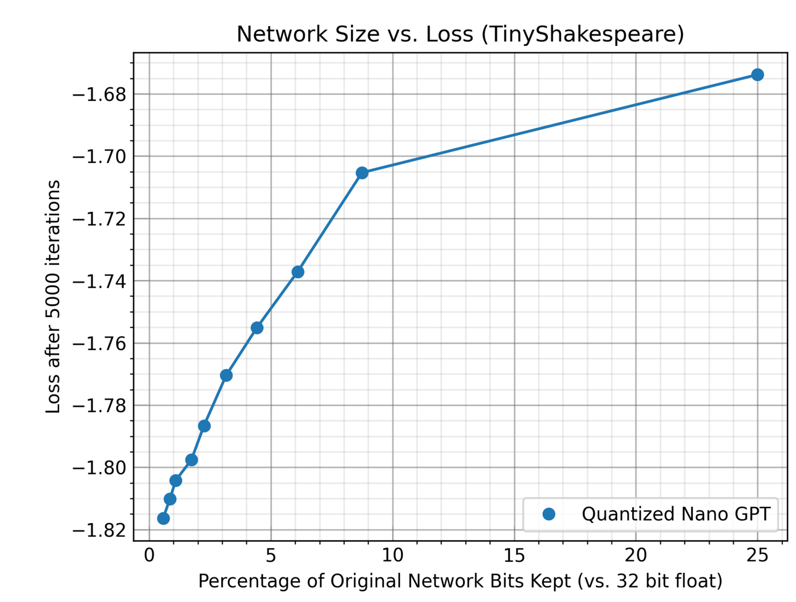


Image classification results from [1,2] and [3]



Scaled down Nano GPT model with 212 K parameters, (from which 131 K parameters are quantized)

Uncompressed model: 7.08 M compressible parameters

Highly compressed model: 0.55 M compressed parameters 7.8% weights remained, 0.60% bits (relative to 32bit float)

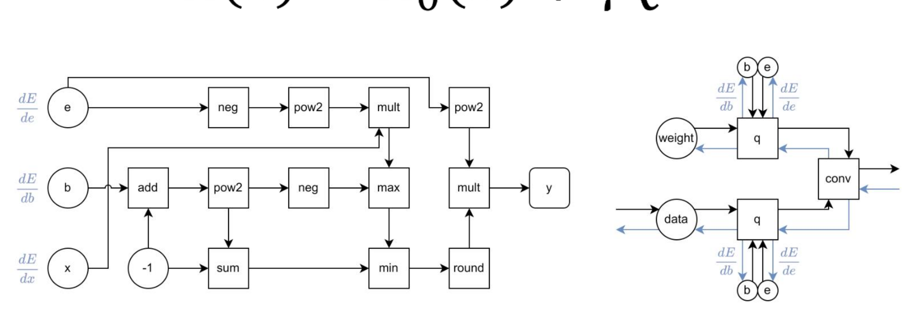
Scaled up model with 10.8 M parameters

METHODOLOGY

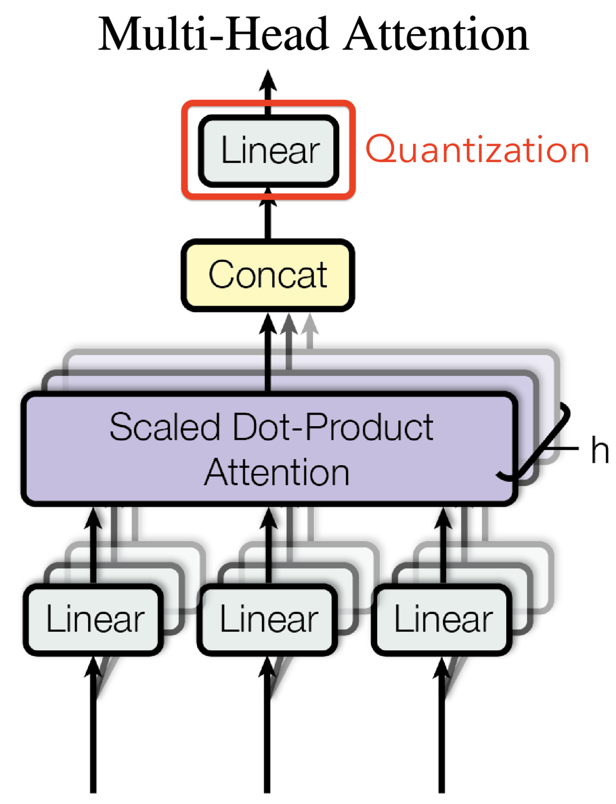
Nano GPT illustration from [6] and [7]

Quantization:

$$\Lambda(x) = \Lambda_0(x) + \gamma Q$$



Forward step and backprogataion of quantization parameters (base and exponent) [1,2]



Quantized part in an attention block [5] implemented in Nano GPT

CONCLUSION

Our demonstration focused on reducing neural network size, which is a major driver of neural network execution time, power consumption, bandwidth, and memory footprint. A key challenge is to reduce size in a manner that can be exploited readily for efficient training and inference without the need for specialized hardware. We applied Self-Compression: a simple, general method that simultaneously achieves two goals: (1) removing redundant weights, and (2) reducing the number of bits required to represent the remaining weights. This was achieved using a generalized loss function to minimize overall network size.

Advantages:
- Fewer weights in the final network;
- Fewer bits in the remaining parameters (depending on the target device);
- Reduced training and execution time;
- Frees the network designer from manually optimizing architectural hyperparameters such as layer widths and bit depths;
- No requirement for special hardware to take advantage of most optimizations (e.g., no need for sparse matrix multiplication or support for hash functions [4]).

Challenges:
- Compressing networks in this way can be challenging. We call one difficulty that we came across *irreversible forgetting*;
- The network is continuously trying to remove ("forget") channels that are not necessary to produce a low error at that moment in training. However, this process could erroneously remove parts of a network that are useful, albeit not heavily used during processing of recent minibatches.

Future work:
- Pruning whole Heads in the Transformer;
- Quantization of Attention layers;
- "Linguistic phase transition" might be detected.

REFERENCES

Our special thanks go to Szabolcs Cséfalvay and Cagatay Dikici. We would also like to thank our other colleagues at Imagination Technologies who supported this work. For suggesting the topic and providing invaluable help Anita Verő is owed special gratitude.

References:

[1] Cséfalvay, Sz.; and Imber, J. Self-Compressing Neural Networks arXiv:2301.13142v2
[2] Cséfalvay, Sz.; Self-Compressing Neural Networks https://blog.imaginationtech.com/self-compressing-neural-networks
[3] Défossez, A.; Adi, Y.; and Synnaeve, G. Differentiable Model Compression via Pseudo Quantization Noise arXiv:2104.09987v3
[4] Han, S.; Mao, H.; and Dally, W. Deep Compression arXiv:1510.00149v5
[5] Vaswani, A.; Shazeer N.; Parmar N.; Uszkoreit J.; Jones L, Gomez A. N.; Kaiser L.; Polosukhin I. Attention Is All You Need arXiv:1706.03762
[6] Karpathy, A. nanoGPT https://github.com/karpathy/nanoGPT
[7] Karpathy, A. Let's build GPT: from scratch, in code, spelled out. https://www.youtube.com/watch?v=kCc8FmEb1nY
[8] Karpathy, A. TinyShakespeare https://huggingface.co/datasets/tiny_shakespeare