NEURAL INFORMATION PROCESSING SYSTEMS

UNIVERSITY OF COPENHAGEN

Characterizing Learning in Deep Neural Networks using Tractable Algorithmic Complexity Analysis

Pedram Bakhtiarifard[1], Sophia N. Wilson[1], Mahmoud H. A. Afifi[2], Jonathan Wenshøj[1], Raghavendra Selvan[1]

[1] University of Copenhagen
[2] The American University in Cairo

Complexity of Binary Objects

Which of the two sequences is most complex?

`A = [101110010110]`

`B = [100100100100]`

Complexity of Binary Objects

Which of the two sequences is most complex?

`A = [101110010110]`

`B = [100100100100]`

`print "101110010110"`

`[print "100"]` $\times 4$

Complexity of Binary Objects

Which of the two sequences is most complex?

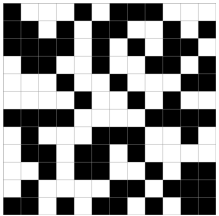
```
A = [101110010110]
```

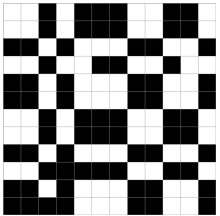
```
print "101110010110"
```

```
B = [100100100100]
```

```
[print "100"] × 4
```

How about the two matrices?





Algorithmic (KCS) Complexity

Algorithmic, or Kolmogorov–Chaitin–Solomonoff (KCS), complexity of a finite object $\mathbf{w} \in \{0,1\}^*$ is the length of the shortest program p among all possible programs that output $\mathbf{w}$ and halt:

$$\mathcal{C}(\mathbf{w}) := \min\Big\{|\mathrm{p}| : U(\mathrm{p}) = \mathbf{w}\Big\}.$$

Here U is a fixed universal prefix-free Turing machine and $|\mathrm{p}|$ denotes the length of the program in bits. The KCS complexity is, however, uncomputable.

Coding Theorem Method (CTM)

The algorithmic probability is

$$P(\mathbf{w}) = \sum_{\mathrm{p}:\, U(\mathrm{p})=\mathbf{w}} 2^{-|\mathrm{p}|}.$$

It relates to $\mathcal{C}$ via the coding theorem

$$\mathcal{C}(\mathbf{w}) = -\log_2 P(\mathbf{w}) + O(1).$$

CTM estimates $P(\mathbf{w})$ by running all n-state Turing machines and taking the fraction D_n that halt with output $\mathbf{w}$:

$$\mathcal{C}_{\mathrm{CTM}}(\mathbf{w}) = -\log_2 D_n(\mathbf{w}) \approx \mathcal{C}(\mathbf{w}).$$

Block Decomposition Method (BDM)

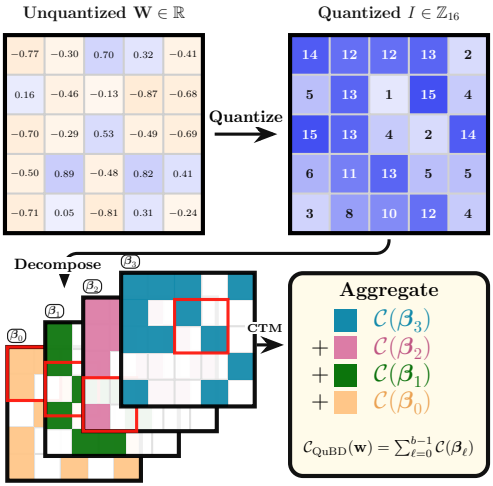
CTM does not scale to large objects. BDM extends it by splitting $\mathbf{w}$ into smaller blocks $\Pi(\mathbf{w}) = (\mathbf{w}_1, \ldots, \mathbf{w}_m)$.

Let $\Pi_u(\mathbf{w})$ be the set of distinct blocks, and let $n_{\mathbf{w}_j}$ count the occurrences of block $\mathbf{w}_j$. Then

$$\mathcal{C}_{\mathrm{BDM}}(\mathbf{w}) = \sum_{\mathbf{w}_j \in \Pi_u(\mathbf{w})} \left(\mathcal{C}_{\mathrm{CTM}}(\mathbf{w}_j) + \log_2 n_{\mathbf{w}_j} \right).$$

This gives a KCS estimate based on the blocks' local bit patterns and their repetition counts.

Quantized Block Decomposition (QuBD) Method



QuBD first quantizes weights into finite codes (top), then decomposes them into aligned bit-planes to aggregate the per-plane QuBD complexities (bottom).

QuBD Complexity

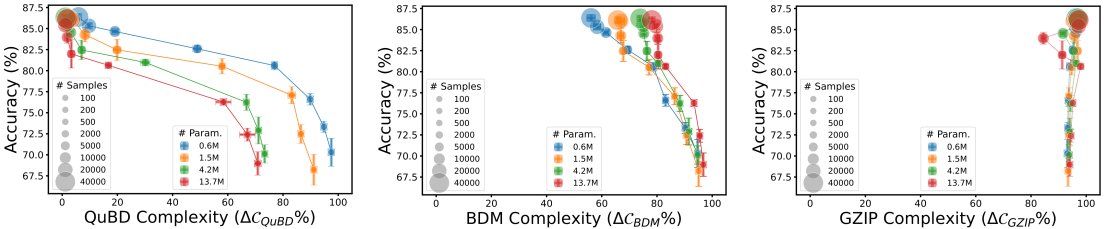
For an object with b bits per symbol, we have b bit-planes $\boldsymbol{\beta}_0, \ldots, \boldsymbol{\beta}_{b-1}$. We sum the BDM complexity estimates across these b planes:

$$\mathcal{C}_{\mathrm{QuBD}}(\mathbf{w}) = \sum_{\ell=0}^{b-1} \mathcal{C}_{\mathrm{BDM}}(\boldsymbol{\beta}_\ell).$$

We examine this complexity and its normalized value relative to a randomized counterpart $\tilde{\mathbf{w}}$:

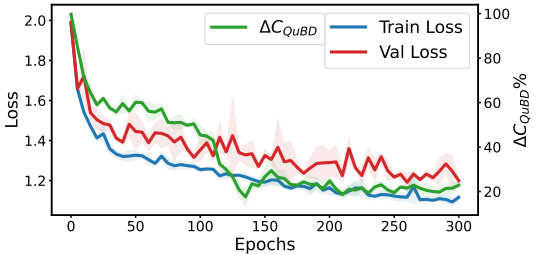
$$\Delta\mathcal{C}_{\mathrm{QuBD}} = \frac{\mathcal{C}_{\mathrm{QuBD}}(\mathbf{w})}{\mathcal{C}_{\mathrm{QuBD}}(\tilde{\mathbf{w}})}.$$

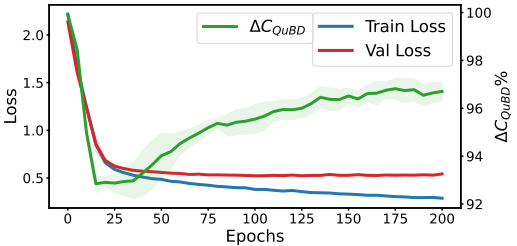
Learning Reduces QuBD Complexity



Accuracy vs. normalized complexities using QuBD, BDM, and GZIP. QuBD complexity tracks the connection between learning and complexity best.

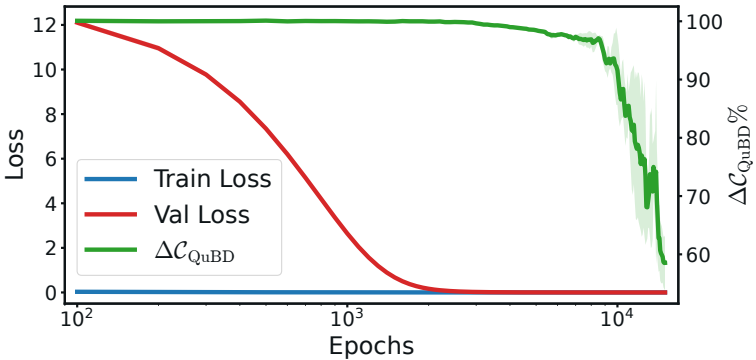
QuBD Complexity Tracks Learning and Overfitting





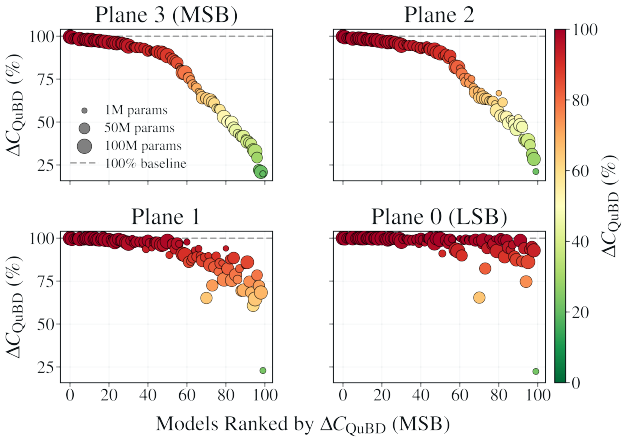
In both cases, QuBD complexity tracks the generalization performance and for MLP it even tracks overfitting.

QuBD Complexity during Grokking



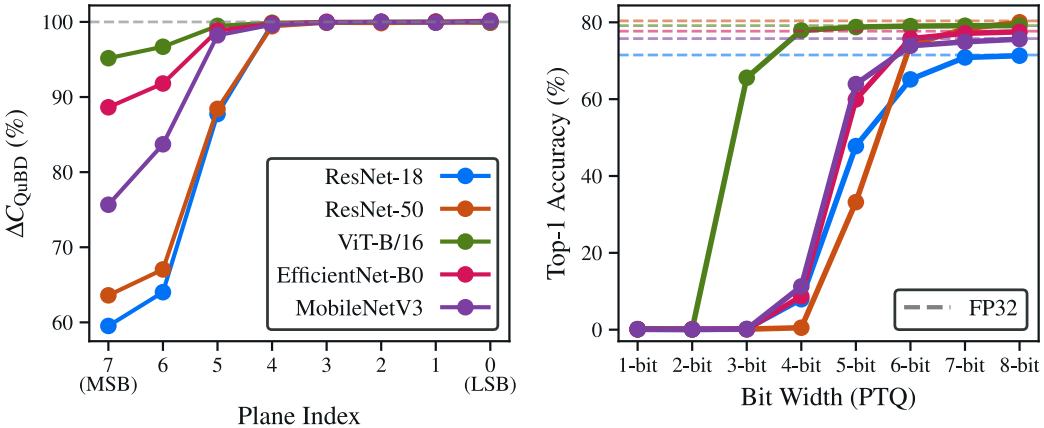
QuBD complexity reduces after the validation loss has reached zero.

Complexity across Bit-planes



Normalized QuBD complexity for 100 pre-trained models with up to 100M parameters with four bit-planes.

QuBD Complexity as a Diagnostic for Compression



Normalized QuBD complexity per bit-plane (left) and top-1 PTQ accuracy (ImageNet-1k).

Paper, Code, and SAINTS Lab



Paper



Code



SAINTS Lab