

# Layered BDM as a Texture and Weighted Network Descriptor to Estimate Kolmogorov Complexity

September 3rd, 2016

Author: Antonio Rueda-Toicen

antonio.rueda.toicen@gmail.com

A layered version of the Block Decomposition Method[1], serves as a descriptor of both weighted networks and grayscale or color images. This descriptor provides an estimate of Kolmogorov Complexity that's sensitive to morphological perturbative [2]. To estimate the complexity of a grayscale texture, we quantize it and aggregate the estimated Kolmogorov complexity values of binary 4 x 4 squares, estimated through the Coding Theorem Method [3, 4].

```
In[4]:= Clear["Global`*"]
SetDirectory[NotebookDirectory[]]
```

```
Out[5]= C:\Users\Antonio Rueda Toicen\Documents\Python\EstimationKolmogorovComplexityImages\
ImageAnalysisWithAlgorithmicInformation
```

```
In[6]:= data = Import["fourByFourCTMs.csv", "CSV", "Numeric" -> False]
```

```
Out[6]= {{0000000000000000, 22.006706292292176},
{0000000000000001, 23.347935957593144}, {0000000000000010, 24.325701071360243},
... 65 530 ... , {1111111111111101, 24.325701071360243},
{1111111111111110, 23.347935957593144}, {1111111111111111, 22.006706292292176}}
```

large output

show less

show more

show all

set size limit...

```
In[7]:= fourByFourCTMs = Transpose@{data[[All, 1]], ToExpression /@ data[[All, 2]]};
```

```
In[8]:= Table[
  (CTM[fourByFourCTMs[[i, 1]]] = fourByFourCTMs[[i, 2]]), {i, 1, 65536, 1}];
```

```
In[9]:= CTM["0000000000000000"]
```

```
Out[9]= 22.0067
```

```
In[10]:= Clear[data, fourByFourCTMs]
```

**“Layered BDM” works through the binary quantization of a texture’s digital levels. The examples below quantize textures using 256 digital levels (byte resolution);  $2^{16}$ ,  $2^{32}$ , etc. quantizations are obviously also possible.**

```

In[11]:= layerDecomposition[image_] :=
  Module[{getLayers, getBlocks, blockCount, stringifiedBlocks},
    getLayers[imag_] := ParallelTable[Unitize[
      ImageData[ColorConvert[imag, "Grayscale"], "Byte"], i], {i, 1, 255, 1}];
    getBlocks[layers_] := Nest[Flatten[#, 1] &,
      Partition[#, {4, 4}, 1] & /@ layers, 2];
    blockCount = Tally[getBlocks[getLayers[image]]];
    stringifiedBlocks =
      StringJoin /@ Map[ToString, (Flatten /@ blockCount[All, 1]), {2}];
    Total[CTM /@ stringifiedBlocks] + Total[Log2[blockCount[All, 2]]]
  ]

```

## Layered BDM as a Texture Descriptor

```

In[12]:= woodTextures =
  (Image[ColorConvert[#, "Grayscale"], "Byte"] & /@ ImageResize[#, {128, 128}] & /@
  (ExampleData[#] & /@
    {"Texture", "Wood"}, {"Texture", "Wood2"}, {"Texture", "Wood3"}));

```

```

In[13]:= smallWT = ImageResize[woodTextures[[1]], {15, 15}]

```

Out[13]=



```

In[14]:= layerDecomposition[smallWT]

```

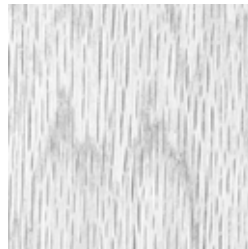
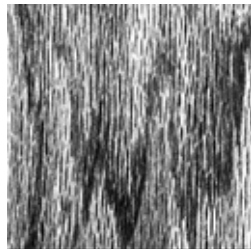
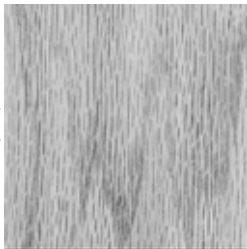
Out[14]= 28 364.8

```

In[15]:= woodTextures

```

Out[15]= {



}

**BDM seems to align with the intuitive notion of a texture's complexity.**

```

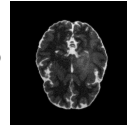
In[16]:= layerDecomposition /@ woodTextures

```

Out[16]= {457 899., 569 941., 398 956.}

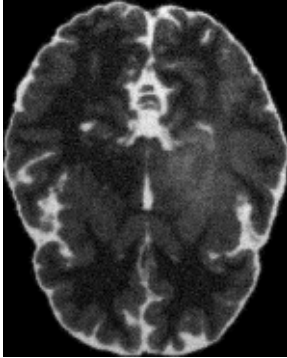
## Test on MR Image with/without an Artifact

```
In[17]:= t2BrainSlice = Image[ColorConvert[ImageCrop@
```



```
, "Grayscale"], "Byte"]
```

Out[17]=



```
In[18]:= t2BrainSlice // ImageData // Dimensions
```

Out[18]= {185, 149}

```
In[19]:= aImage = ColorNegate@
          ColorConvert[Image[Rasterize[Style["a", FontSize → 50]], "Byte"], "Grayscale"]
```

Out[19]=



```
In[20]:= brainWLetter =
          Image[ColorConvert[ImageAdd[t2BrainSlice, aImage], "Grayscale"], "Byte"]
```

Out[20]=



```
In[21]:= brainWLetter // ImageData // Dimensions
```

Out[21]= {185, 149}

The image with a simple texture artifact has lower BDM and lower texture complexity.

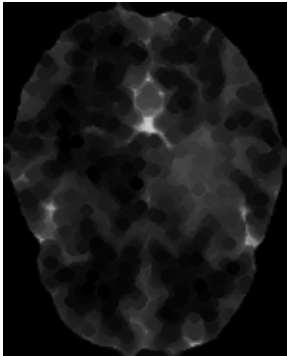
```
In[22]:= layerDecomposition /@ {t2BrainSlice, brainWLetter}
```

Out[22]=  $\{1.39659 \times 10^6, 1.38208 \times 10^6\}$

## Sensitivity to Morphological Perturbation

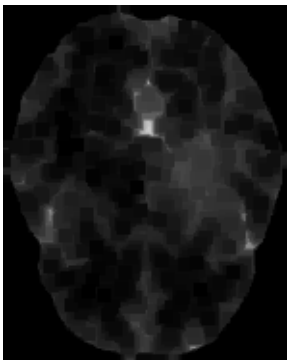
```
In[23]:= t2BrainSliceWDiskErosion = Erosion[t2BrainSlice, DiskMatrix[3]]
```

Out[23]=



```
In[24]:= t2BrainSliceWBoxedErosion = Erosion[t2BrainSlice, BoxMatrix[3]]
```

Out[24]=



LayeredBDM is highly sensitive to morphological perturbations of the data.

```
In[25]:= layerDecomposition /@ {t2BrainSliceWBoxedErosion, t2BrainSliceWDiskErosion}
```

Out[25]= {59 706.4, 66 779.8}

## Layered BDM as a Weighted Network Descriptor

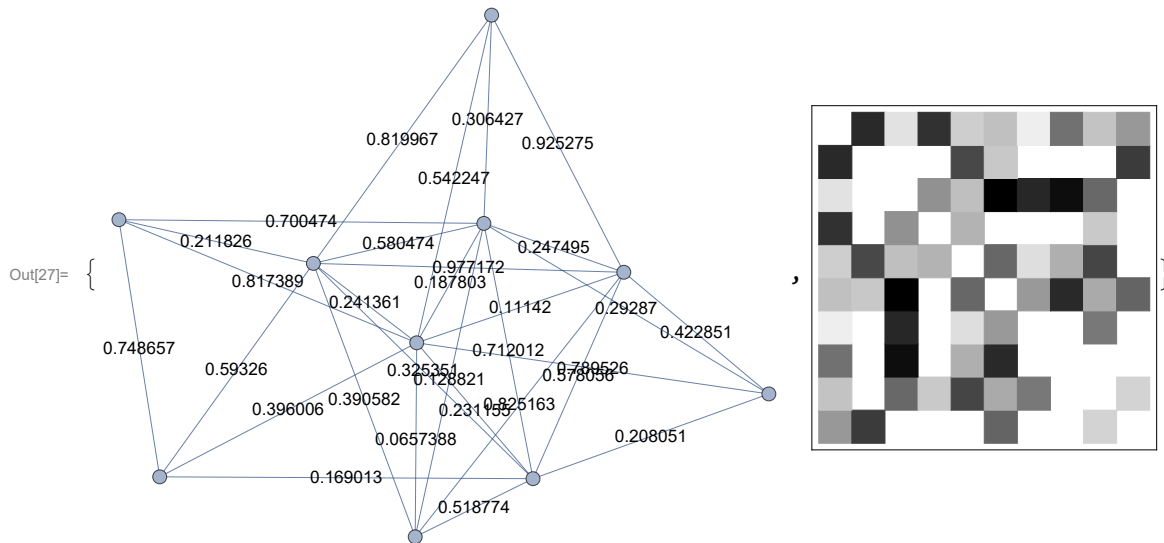
**BDM can be used to evaluate networks trained through backpropagation.**

```
In[26]:= SeedRandom[1]; rg = RandomGraph[{10, 30}];
```

```

In[27]:= SeedRandom[1];
{wrg = Graph[EdgeList[rg], EdgeWeight → RandomReal[1, Length[EdgeList[rg]]],
  EdgeLabels → "EdgeWeight", ImageSize → Medium],
  ArrayPlot@WeightedAdjacencyMatrix[wrg]}

```



```

In[28]:= floatToByte[float_] := Floor[If[float == 1.0, 255, float * 256.0]]

```

```

In[29]:= floatToByte /@ {0.001, 0.999}

```

```

Out[29]:= {0, 255}

```

```

In[30]:= byteWeightedMatrix = Map[floatToByte, Normal[WeightedAdjacencyMatrix[wrg]], {2}];
MatrixForm[byteWeightedMatrix]

```

```

Out[30]//MatrixForm=
(
  0  209  28  202  48  61  16  138  59  101
  209  0  0  0  179  54  0  0  0  191
  28  0  0  108  63  250  211  236  147  0
  202  0  108  0  74  0  0  0  53  0
  48  179  63  74  0  148  32  78  182  0
  61  54  250  0  148  0  99  209  83  151
  16  0  211  0  32  99  0  0  132  0
  138  0  236  0  78  209  0  0  0  0
  59  0  147  53  182  83  132  0  0  43
  101  191  0  0  0  151  0  0  43  0
)

```

```

In[31]:= layerDecompositionForWeightedGraphs[weightedGraph_] :=
Module[{floatToByte, getLayers, getBlocks,
  blockCount, stringifiedBlocks, weightedAdjMatrix},
  floatToByte[float_] := Floor[If[float == 1.0, 255, float * 256.0]];
  getLayers[w_] := ParallelTable[
    Unitize[ Map[floatToByte, weightedAdjMatrix, {2}], i], {i, 1, 255, 1}];
  getBlocks[layers_] := Nest[Flatten[#, 1] &,
    Partition[#, {4, 4}, 1] & /@ layers, 2];
  weightedAdjMatrix = Normal[WeightedAdjacencyMatrix[weightedGraph]];
  blockCount = Tally[getBlocks[getLayers[weightedAdjMatrix]]];
  stringifiedBlocks =
    StringJoin /@ Map[ToString, (Flatten /@ blockCount[[All, 1]]), {2}];
  Total[CTM /@ stringifiedBlocks] + Total[Log2[blockCount[[All, 2]]]]
]

```

```
In[32]:= layerDecompositionForWeightedGraphs [wrg]
```

```
Out[32]= 12 323.2
```

---

## References

- [1] Antonio Rueda-Toicen, “Morphological Image Analysis through Estimations of Kolmogorov Complexity” (in preparation)
- [2] Hector Zenil, Santiago Hernández - Orozco, Narsis A.Kiani, Fernando Soler - Toscano, Antonio Rueda - Toicen, and Jesper Tegner "A Decomposition Method for Global Evaluation of Shannon Entropy and Local Estimations of Algorithmic Complexity", <https://arxiv.org/abs/1609.00110>
- [3] Fernando Soler - Toscano, Hector Zenil, Jean-Paul Delahaye, and Nicolas Gauvrit (2014) “Calculating Kolmogorov Complexity from the Output Frequency Distributions of Small Turing Machines.” PLoS ONE 9 (5) : e96223.