New CA Classifiers (random colours)

Wolfram Classes of ECAs

In[1]:=
```
CAclasses = {1, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 3,
   2, 2, 2, 2, 2, 2, 2, 3, 2, 1, 2, 2, 2, 2, 2, 2, 2, 1, 3, 2, 2, 2, 3, 2, 2, 2,
   2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 3, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
   3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 3, 3, 2, 2, 2, 2, 2, 1, 3, 2, 2, 2,
   3, 3, 2, 2, 3, 3, 3, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 2, 4, 2, 3,
   2, 1, 3, 2, 2, 2, 2, 2, 3, 1, 4, 2, 2, 2, 2, 2, 2, 2, 2, 3, 4, 2, 3, 3, 3, 2,
   3, 2, 2, 2, 2, 2, 2, 1, 3, 2, 2, 2, 3, 2, 2, 1, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2,
   2, 2, 2, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 1, 4, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2,
   2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2,
   2, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1};
```

Functions for creating net and random datasets (ECAs, all 4 classes)

In[2]:=
```
RandomRuleC[n_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[n, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame → False]]
netC[W_Integer, H_Integer] := NetInitialize@
  NetChain[{ConvolutionLayer[16, {2, 3}], Ramp, PoolingLayer[{H, W} - {1, 2}],
    FlattenLayer[], LinearLayer[256], SoftmaxLayer[]},
   "Input" → NetEncoder[{"Image", {W, H}}],
   "Output" → NetDecoder[{"Class", Range[0, 255]}]]
netTwoCC[W_Integer, H_Integer] := NetInitialize@
  NetChain[<|"conv1" → ConvolutionLayer[16, {2, 3}], "ramp1" → Ramp,
    "conv3" → ConvolutionLayer[16, {2, 3}], "ramp2" → Ramp,
    "pooling" → PoolingLayer[{H, W} - {2, 4}], "flatten" → FlattenLayer[],
    "linear" → 512, "linear2" → 4, "softmax" → SoftmaxLayer[]|>,
   "Input" → NetEncoder[{"Image", {W, H}}],
   "Output" → NetDecoder[{"Class", Range[1, 4]}]]
dataC[W_Integer, H_Integer, n_Integer] := Table[
  RandomRuleC[i, W, H] → CAclasses[[i + 1]], {i, RandomChoice[Range[0, 255], n]}]
```

In[6]:=
```
netThreeCC[W_Integer, H_Integer] :=
 NetInitialize@NetChain[<|"conv1" → ConvolutionLayer[16, {2, 3}],
    "ramp1" → Ramp, "conv2" → ConvolutionLayer[16, {2, 3}],
    "ramp2" → Ramp, "conv3" → ConvolutionLayer[16, {2, 3}], "ramp3" → Ramp,
    "pooling" → PoolingLayer[{H, W} - {4, 8}], "flatten" → FlattenLayer[],
    "linear" → 512, "linear2" → 4, "softmax" → SoftmaxLayer[]|>,
   "Input" → NetEncoder[{"Image", {W, H}}],
   "Output" → NetDecoder[{"Class", Range[1, 4]}]]
```

In[7]:=
```
netThreeCC1024[W_Integer, H_Integer] :=
 NetInitialize@NetChain[<|"conv1" → ConvolutionLayer[16, {2, 3}],
    "ramp1" → Ramp, "conv2" → ConvolutionLayer[16, {2, 3}],
    "ramp2" → Ramp, "conv3" → ConvolutionLayer[16, {2, 3}], "ramp3" → Ramp,
    "pooling" → PoolingLayer[{H, W} - {4, 8}], "flatten" → FlattenLayer[],
    "linear" → 1024, "linear2" → 4, "softmax" → SoftmaxLayer[]|>,
   "Input" → NetEncoder[{"Image", {W, H}}],
   "Output" → NetDecoder[{"Class", Range[1, 4]}]]
```

In[8]:=
```
netFourCC512[W_Integer, H_Integer] :=
 NetInitialize@NetChain[<|"conv1" → ConvolutionLayer[32, {2, 3}],
    "ramp1" → Ramp, "conv3" → ConvolutionLayer[32, {2, 3}], "ramp2" → Ramp,
    "pooling" → PoolingLayer[{H, W} - {2, 4}], "flatten" → FlattenLayer[],
    "linear" → 512, "linear2" → 4, "softmax" → SoftmaxLayer[]|>,
   "Input" → NetEncoder[{"Image", {W, H}}],
   "Output" → NetDecoder[{"Class", Range[1, 4]}]]
```

In[9]:=
```
netFiveCC512[W_Integer, H_Integer] :=
 NetInitialize@NetChain[<|"conv1" → ConvolutionLayer[32, {2, 3}],
    "bat1" → BatchNormalizationLayer[], "ramp1" → Ramp,
    "conv3" → ConvolutionLayer[32, {2, 3}],
    "bat2" → BatchNormalizationLayer[], "ramp2" → Ramp,
    "pooling" → PoolingLayer[{H, W} - {2, 4}], "flatten" → FlattenLayer[],
    "linear" → 512, "linear2" → 4, "softmax" → SoftmaxLayer[]|>,
   "Input" → NetEncoder[{"Image", {W, H}}],
   "Output" → NetDecoder[{"Class", Range[1, 4]}]]
```

In[10]:=
```
netSixCC512drop[W_Integer, H_Integer] :=
 NetInitialize@NetChain[<|"drop1" → DropoutLayer[0.2], "conv1" →
     ConvolutionLayer[32, {3, 3}], "bat1" → BatchNormalizationLayer[],
    "ramp1" → Ramp, "conv3" → ConvolutionLayer[32, {3, 3}],
    "bat2" → BatchNormalizationLayer[], "ramp2" → Ramp,
    "pooling" → PoolingLayer[{H, W} - {4, 8}], "flatten" → FlattenLayer[],
    "linear" → 512, "drop2" → DropoutLayer[0.2], "linear2" → 4,
    "softmax" → SoftmaxLayer[]|>, "Input" → NetEncoder[{"Image", {W, H}}],
   "Output" → NetDecoder[{"Class", Range[1, 4]}]]
```

In[11]:=
```
netSevenCC512drop[W_Integer, H_Integer] :=
 NetInitialize@NetChain[<|"conv1" → ConvolutionLayer[24, {3, 3}],
    "bat1" → BatchNormalizationLayer[], "ramp1" → Ramp,
    "conv3" → ConvolutionLayer[24, {3, 3}],
    "bat2" → BatchNormalizationLayer[], "ramp2" → Ramp,
    "pooling" → PoolingLayer[{H, W} - {4, 8}], "flatten" → FlattenLayer[],
    "linear" → 512, "drop2" → DropoutLayer[0.2], "linear2" → 4,
    "softmax" → SoftmaxLayer[]|>, "Input" → NetEncoder[{"Image", {W, H}}],
   "Output" → NetDecoder[{"Class", Range[1, 4]}]]
```

In[12]:=
```
netEightCC512drop[W_Integer, H_Integer] :=
 NetInitialize@NetChain[<|"conv1" → ConvolutionLayer[24, {3, 3}],
    "bat1" → BatchNormalizationLayer[], "ramp1" → Ramp,
    "conv2" → ConvolutionLayer[16, {2, 3}], "bat2" → BatchNormalizationLayer[],
    "ramp2" → Ramp, "conv3" → ConvolutionLayer[24, {3, 3}],
    "bat3" → BatchNormalizationLayer[], "ramp3" → Ramp,
    "pooling" → PoolingLayer[{H, W} - {8, 16}], "flatten" → FlattenLayer[],
    "linear" → 1024, "drop2" → DropoutLayer[0.2], "linear2" → 4,
    "softmax" → SoftmaxLayer[]|>, "Input" → NetEncoder[{"Image", {W, H}}],
   "Output" → NetDecoder[{"Class", Range[1, 4]}]]
```

In[13]:=
```
netNineCC512drop[W_Integer, H_Integer] :=
 NetInitialize@NetChain[<|"conv1" → ConvolutionLayer[24, {3, 3}],
    "bat1" → BatchNormalizationLayer[], "ramp1" → Ramp,
    "conv2" → ConvolutionLayer[24, {3, 3}], "bat2" → BatchNormalizationLayer[],
    "ramp2" → Ramp, "conv3" → ConvolutionLayer[24, {3, 3}],
    "bat3" → BatchNormalizationLayer[], "ramp3" → Ramp,
    "pooling" → PoolingLayer[{H, W} - {12, 12}], "flatten" → FlattenLayer[],
    "linear" → 512, "drop2" → DropoutLayer[0.2], "linear2" → 4,
    "softmax" → SoftmaxLayer[]|>, "Input" → NetEncoder[{"Image", {W, H}}],
   "Output" → NetDecoder[{"Class", Range[1, 4]}]]
```

Functions for creating datasets (1D totalistic CAs)

k=3, r=1 totalistic (class 4 only)

In[14]:=
```
gen3TC[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {3, 1}}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
data3T2C[W_Integer, H_Integer, n_Integer] := Table[gen3TC[i, W, H] → 4,
  {i, RandomChoice[{1635, 1815, 2007, 2043, 2049, 1388, 1041}, n]}]
```

k=4, r=1 totalistic (class 4 only, 1 example)

In[16]:=
```
gen4TC[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {4, 1}}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
data4TC[W_Integer, H_Integer, n_Integer] := Table[gen4TC[1 004 600, W, H] → 4, n]
```

k=2, r=2 totalistic (all 4 classes)

In[18]:=
```
gen2r2C[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {2, 1}, 2}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
data2r2c4C[W_Integer, H_Integer, n_Integer] :=
 Table[gen2r2C[i, W, H] → 4, {i, RandomChoice[{20, 52}, n]}]
data2r2c3C[W_Integer, H_Integer, n_Integer] :=
 Table[gen2r2C[i, W, H] → 3, {i, RandomChoice[
    {2, 6, 10, 12, 14, 18, 22, 26, 28, 30, 34, 38, 42, 44, 46, 50}, n]}]
data2r2c2C[W_Integer, H_Integer, n_Integer] :=
 Table[gen2r2C[i, W, H] → 2, {i, RandomChoice[{8, 24, 56}, n]}]
data2r2c1C[W_Integer, H_Integer, n_Integer] := Table[gen2r2C[i, W, H] → 1,
  {i, RandomChoice[{0, 4, 16, 32, 36, 40, 48, 54, 58, 60, 62}, n]}]
genData2r2C[W_Integer, H_Integer, n_Integer] := Join[data2r2c4C[W, H, n],
  data2r2c3C[W, H, n], data2r2c2C[W, H, n], data2r2c1C[W, H, n]]
```

k=5, r=1 totalistic (class 4 only)

In[24]:=
```
gen5T4C[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {5, 1}}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
data5T4C[n_Integer, W_Integer, H_Integer] :=
 Table[gen5T4C[i, W, H] → 4, {i, RandomChoice[
    {781 130 654, 772 514 435, 1 151 319 452, 309 095 787, 880 862 046, 973 835 714,
     779 446 817, 345 466 505, 535 500 975, 793 363 571, 1 052 373 865, 455 984 785,
     339 227 109, 1 050 973 846, 513 368 817, 91 315 820, 113 925 357}, n]}]
```

k=5, r=1 totalistic (classes 2/3/4)

In[26]:=
```
gen5TC[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {5, 1}, 1}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
data5T4CC[W_Integer, H_Integer, n_Integer] := Table[gen5TC[i, W, H] → 4,
  {i, RandomChoice[{644 218 533, 491 739 943, 6 889 640, 986 144 962, 1 099 816 682,
     988 971 204, 300 829 994, 272 622 024, 304 100 638, 626 595 633}, n]}]
data5T3CC[W_Integer, H_Integer, n_Integer] := Table[gen5TC[i, W, H] → 3,
  {i, RandomChoice[{889 082 395, 541 068 260, 807 907 479, 816 180 062, 650 485 139,
     643 827 745, 753 940 864, 871 525 323, 351 440 311, 83 501 460}, n]}]
data5T2CC[W_Integer, H_Integer, n_Integer] :=
 Table[gen5TC[i, W, H] → 2, {i, RandomChoice[
    {525 735 659, 1 022 330 944, 1 007 796 739, 495 633 437, 1 036 827 943}, n]}]
genData5TCC[W_Integer, H_Integer, n_Integer] :=
 Join[data5T4CC[W, H, n], data5T3CC[W, H, n], data5T2CC[W, H, n]]
```

Generate test datasets

k=2, r=2 non-totalistic

In[31]:=
```
genk2r2C[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, 2, 2}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
datak2r2C[W_Integer, H_Integer, n_Integer] :=
 Table[genk2r2C[i, W, H] → i, {i, RandomChoice[Range[0, 4 294 967 295], n]}]
```

k=2, r=3 non-totalistic

In[75]:=
```
genk2r3NT[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, 2, 3}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
datak2r3NT[W_Integer, H_Integer, n_Integer] :=
 Table[genk2r3NT[i, W, H] → i, {i, RandomInteger[2^2^7 - 1, n]}]
```

k=3, r=1 non-totalistic

In[87]:=
```
genk3r1NT[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, 3}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
datak3r1NT[W_Integer, H_Integer, n_Integer] :=
 Table[genk3r1NT[i, W, H] → i, {i, RandomInteger[3^3^3 - 1, n]}]
```

k=3, r=2 totalistic

In[35]:=
```
genk3r2C[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {3, 1}, 2}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
datak3r2C[W_Integer, H_Integer, n_Integer] :=
 Table[genk3r2C[i, W, H] → i, {i, RandomChoice[Range[0, 177146], n]}]
```

k=3, r=3 totalistic

In[37]:=
```
genk3r3C[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {3, 1}, 3}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
datak3r3C[W_Integer, H_Integer, n_Integer] :=
 Table[genk3r3C[i, W, H] → i, {i, RandomChoice[Range[0, 14348906], n]}]
```

k=4, r=1 non-totalistic

In[91]:=
```
genk4r1NT[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, 4}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
datak4r1NT[W_Integer, H_Integer, n_Integer] :=
 Table[genk4r1NT[i, W, H] → i, {i, RandomInteger[4^4^3 - 1, n]}]
```

k=4, r=1 totalistic

In[39]:=
```
genk4r1C[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {4, 1}}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
datak4r1C[W_Integer, H_Integer, n_Integer] :=
 Table[genk4r1C[i, W, H] → i, {i, RandomChoice[Range[0, 1048575], n]}]
```

k=4, r=2 totalistic

In[41]:=
```
genk4r2C[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {4, 1}, 2}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
datak4r2C[W_Integer, H_Integer, n_Integer] :=
 Table[genk4r2C[i, W, H] → i, {i, RandomChoice[Range[0, 4294967295], n]}]
```

k=5, r=1 totalistic

In[43]:=
```
gen5T2C[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {5, 1}, 1}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
data5T2C[n_Integer, W_Integer, H_Integer] := Table[gen5T2C[i, W, H] → i,
  {i, RandomChoice[Range[0, 1220703125], n]}]
```

k=6, r=1 totalistic

In[45]:=
```
gen6TC[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {6, 1}, 1}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
data6TC[n_Integer, W_Integer, H_Integer] := Table[gen6TC[i, W, H] → i,
  {i, RandomInteger[2821109907455, n]}]
```

k=6, r=2 totalistic

In[47]:=
```
gen6T2C[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {6, 1}, 2}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame -> False]]
data6T2C[n_Integer, W_Integer, H_Integer] := Table[gen6T2C[i, W, H] → i,
  {i, RandomInteger[170 581 728 179 578 208 255, n]}]
```

k=7, r=1 totalistic

In[49]:=
```
gen7TC[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {7, 1}, 1}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame → False]]
data7TC[n_Integer, W_Integer, H_Integer] := Table[gen7TC[i, W, H] → i,
  {i, RandomInteger[11 398 895 185 373 142, n]}]
```

k=8, r=1 totalistic

In[51]:=
```
gen8TC[p_Integer, W_Integer, H_Integer] :=
 Image[ArrayPlot[CellularAutomaton[{p, {8, 1}, 1}, RandomInteger[1, W], H - 1],
   ImageSize → {W, H}, ColorRules → {0 -> RandomColor[], 1 -> RandomColor[],
     3 -> RandomColor[], 4 → RandomColor[], 5 -> RandomColor[],
     6 -> RandomColor[], 7 -> RandomColor[]}, Frame → False]]
data8TC[n_Integer, W_Integer, H_Integer] := Table[gen8TC[i, W, H] → i,
  {i, RandomInteger[73 786 976 294 838 206 463, n]}]
```

Network XIII - Two convolutions, dropout on linear only, BatchNorm

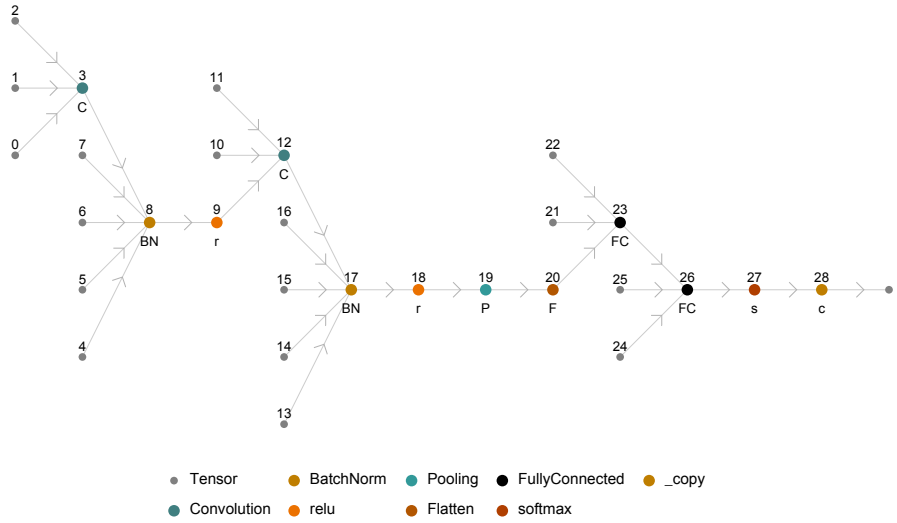
In[•]:=
```
netECA13 = netSevenCC512drop[128, 128]
```

Out[•]=
```
NetChain[
```


]

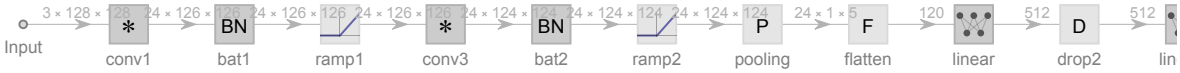
In[•]:=
```
NetInformation[netECA13, "MXNetNodeGraphPlot"]
```

Out[•]=



In[•]:=
```
NetInformation[netECA13, "SummaryGraphic"]
```

Out[•]=



In[•]:=
```
dataECA13 = dataC[128, 128, 8192];
```

In[•]:=
```
dataTotalistic2BigC13 = genData2r2C[128, 128, 1024];
```

In[•]:=
```
dataTotalistic3BigC13 = data3T2C[128, 128, 1024];
```

In[•]:=
```
dataTotalistic4BigC13 = data4TC[128, 128, 1024];
```

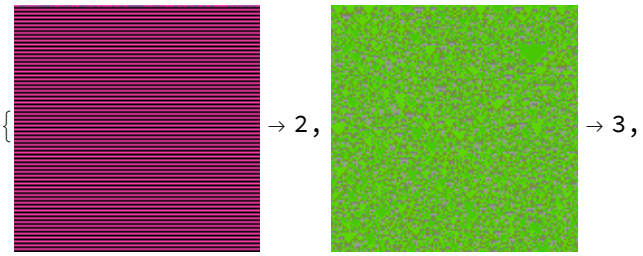
In[•]:=
```
dataTotalistic5BigC13 = genData5TCC[128, 128, 4096];
```

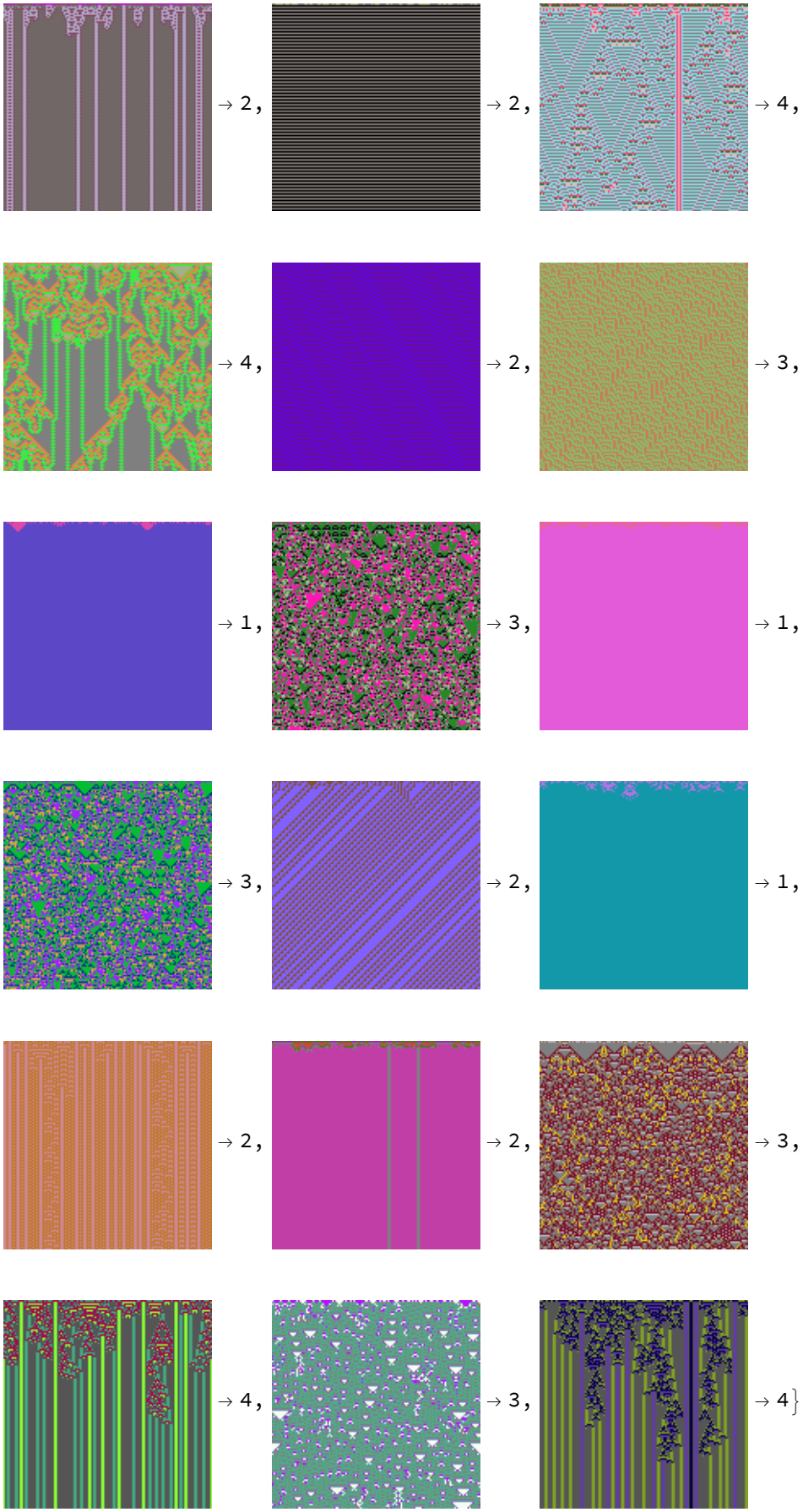
In[•]:=
```
fullTrainingBigC13 = Join[dataECA13, dataTotalistic2BigC13,
   dataTotalistic3BigC13, dataTotalistic4BigC13, dataTotalistic5BigC13];
Length[fullTrainingBigC13]
```

Out[•]=
```
26 624
```

In[•]:=
```
RandomSample[fullTrainingBigC13, 20]
```

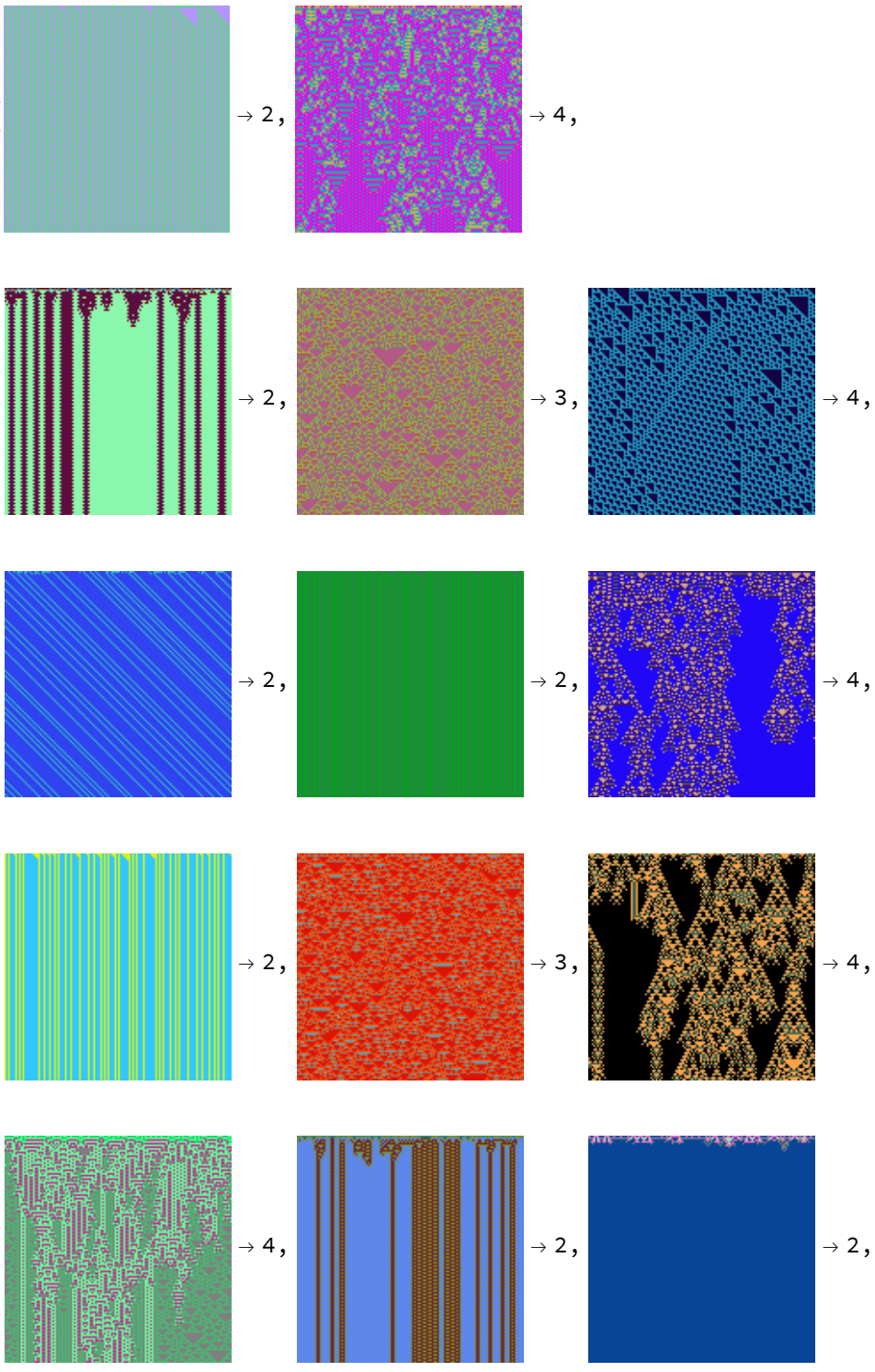
Out[•]=

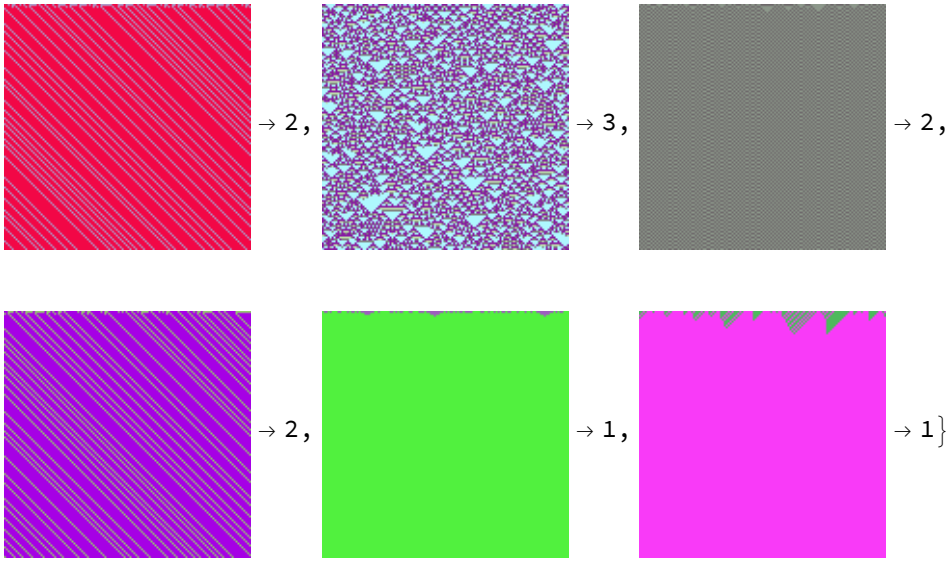



→ 2,
→ 2,
→ 4,
→ 4,
→ 2,
→ 3,
→ 1,
→ 3,
→ 1,
→ 3,
→ 2,
→ 1,
→ 2,
→ 2,
→ 3,
→ 4,
→ 3,
→ 4}

In[•]:= `RandomSample[fullTrainingBigC13, 20]`

Out[•]=





In[•]:=
```
dir = SetDirectory[NotebookDirectory[]]
```

Out[•]=
```
/Users/thorsilver/Downloads/Wolfram notebooks
```

In[•]:=
```
netECA12 = Import["netECA12-r12.wlnet"]
```

Out[•]=
NetChain[

	Input	image array (size: 3 × 120 × 200)
conv1	ConvolutionLayer	array (size: 24 × 118 × 198)
bat1	BatchNormalizationLayer	array (size: 24 × 118 × 198)
ramp1	Ramp	array (size: 24 × 118 × 198)
conv3	ConvolutionLayer	array (size: 24 × 116 × 196)
bat2	BatchNormalizationLayer	array (size: 24 × 116 × 196)
ramp2	Ramp	array (size: 24 × 116 × 196)
pooling	PoolingLayer	array (size: 24 × 1 × 5)
flatten	FlattenLayer	vector (size: 120)
linear	LinearLayer	vector (size: 512)
drop2	DropoutLayer	vector (size: 512)
linear2	LinearLayer	vector (size: 4)
softmax	SoftmaxLayer	vector (size: 4)
	Output	class

]

In[•]:=
```
netECA13 = NetTrain[netECA13, fullTrainingBigC13,
  MaxTrainingRounds → 20, BatchSize → 256 * 4, TargetDevice → "CPU",
  TrainingProgressCheckpointing → {"Directory", dir}]
```

Out[•]=
NetChain[

Input port:	image
Output port:	class
Number of layers:	12

]

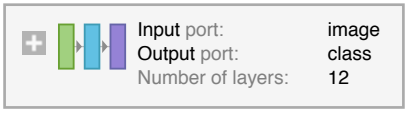
In[•]:=
```
netECA13 = Import["netECA13-r20.wlnet"]
```

Out[•]=
NetChain[

Input port:	image
Output port:	class
Number of layers:	12

]

```
In[•]:= netECA13 = NetTrain[netECA13, fullTrainingBigC13,
          MaxTrainingRounds → 20, BatchSize → 256 * 4, TargetDevice → "CPU",
          TrainingProgressCheckpointing → {"Directory", dir}]
```

Out[•]= NetChain[

]

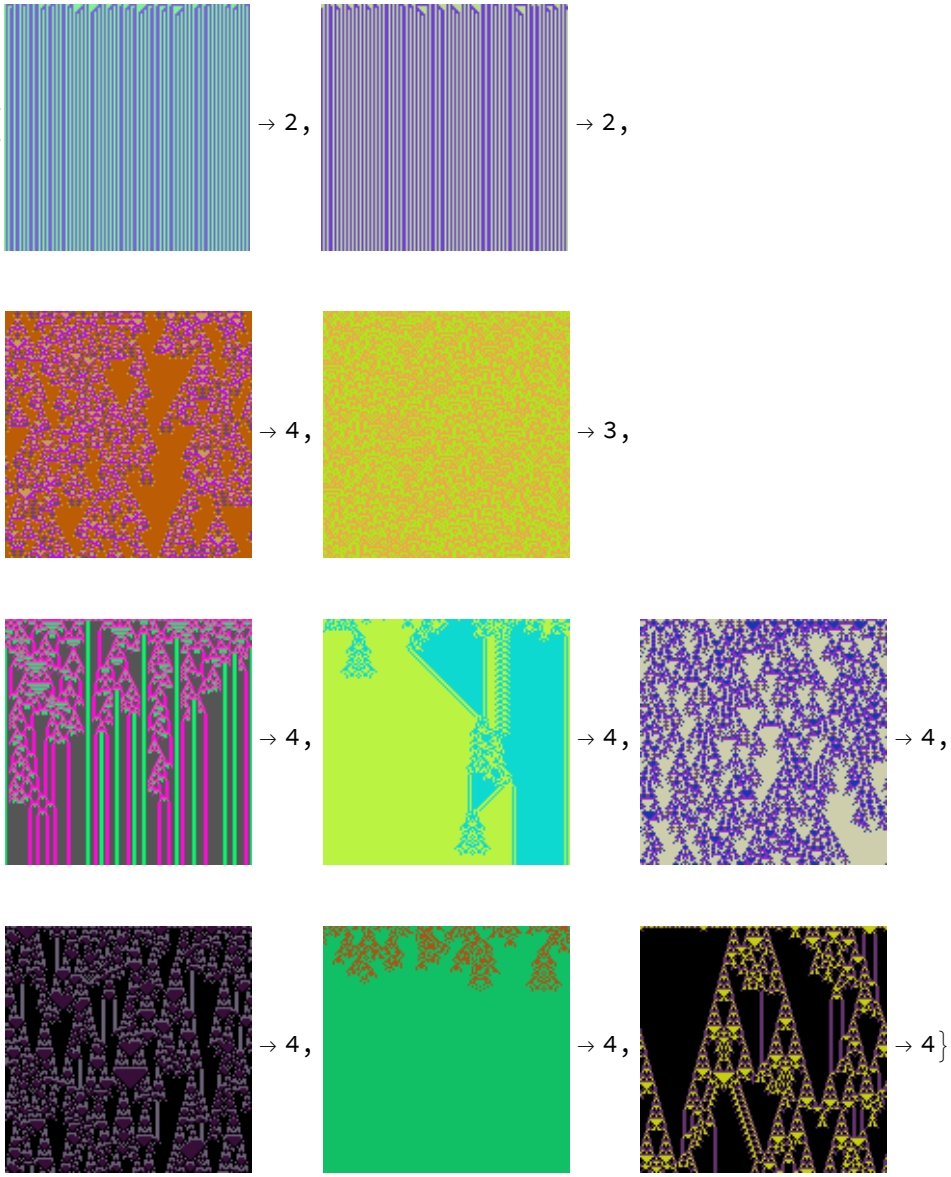
Generate test data for Network XIII

```
In[•]:= testDataECABigC = dataC[128, 128, 1024];
        testData2TBigC = genData2r2C[128, 128, 1024];
        testData3TBigC = data3T2C[128, 128, 1024];
        testData4TBigC = data4TC[128, 128, 1024];
        testData5TBigC = genData5TCC[128, 128, 1024];
        fullTestSetBigC = Join[testDataECABigC,
           testData2TBigC, testData3TBigC, testData4TBigC, testData5TBigC];
        Length[fullTestSetBigC]
```

Out[•]= 10 240

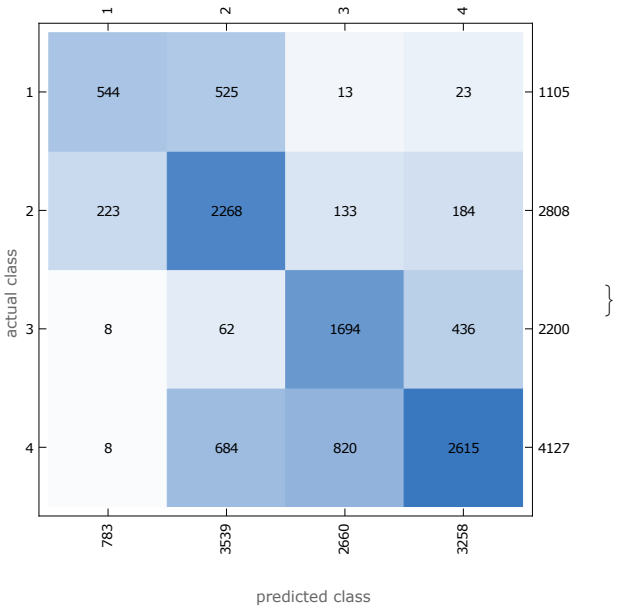
In[•]:= `RandomSample[fullTestSetBigC, 10]`

Out[•]=



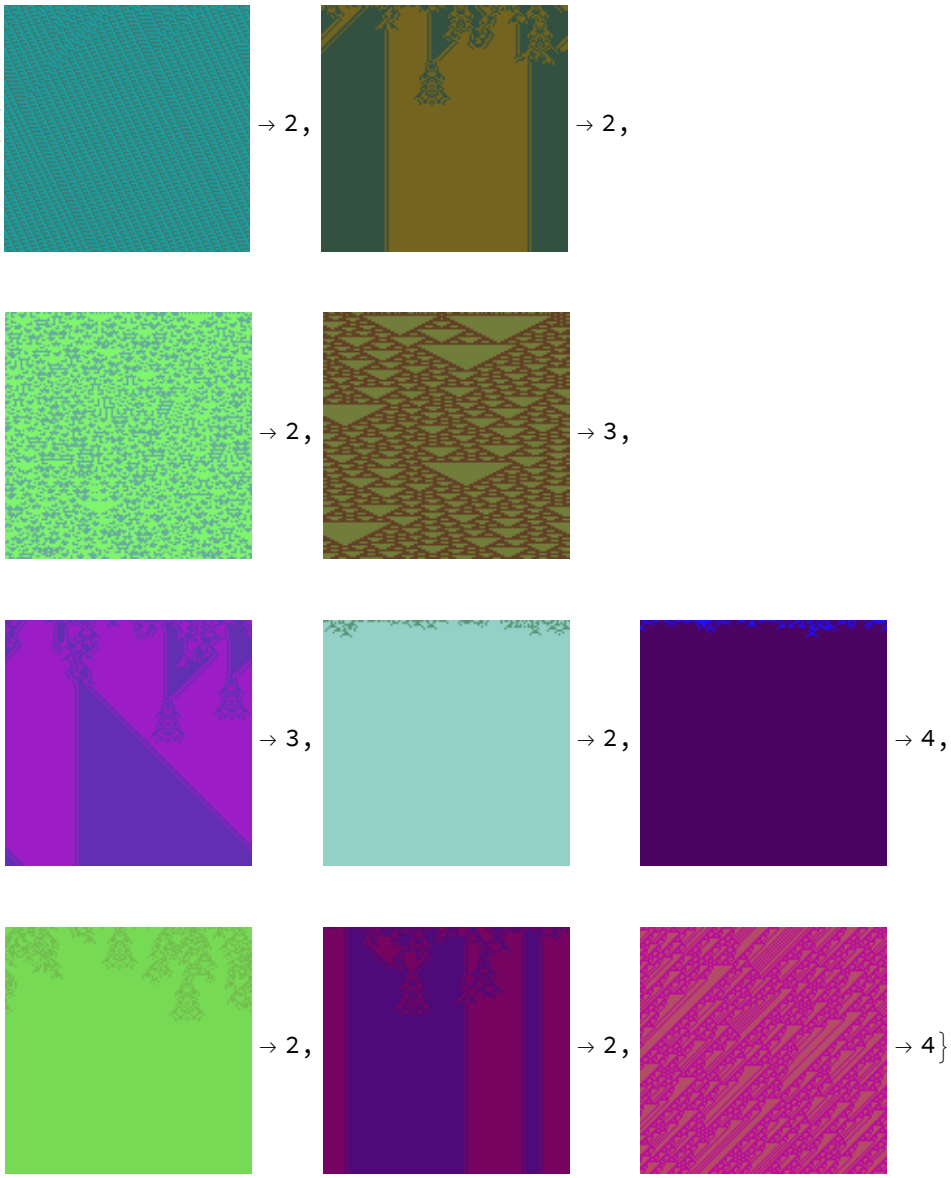
In[]:=
```
NetMeasurements[netECA13, fullTestSetBigC,
 {"Accuracy", "Precision", "ConfusionMatrixPlot"}]
```

Out[]=
```
{0.69541, ⟨|1 → 0.694764, 2 → 0.640859, 3 → 0.636842, 4 → 0.80264|⟩,
```

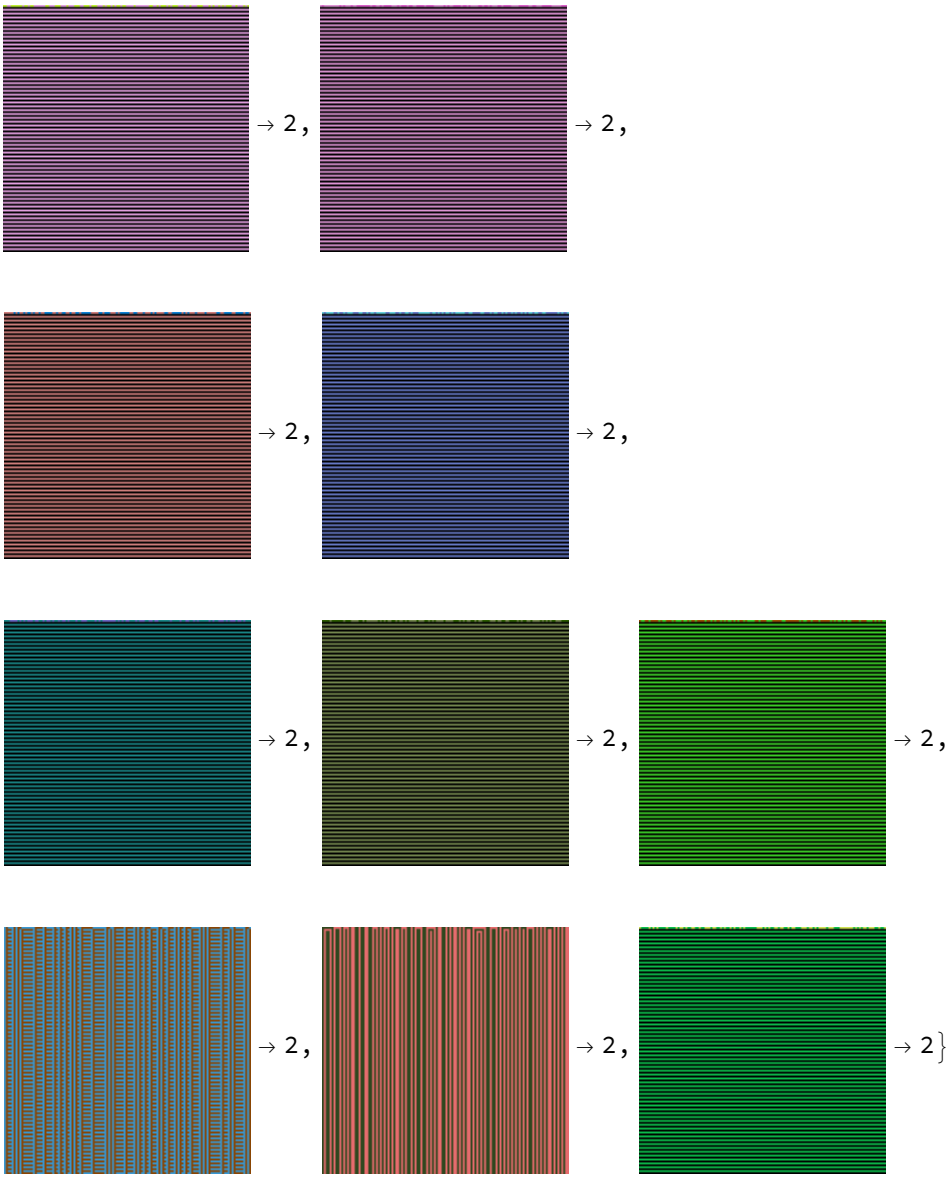


In[]:=
```
entropyImagesBigC = RandomSample[Keys[fullTestSetBigC], 500];
entropiesBigC = netECA13[entropyImagesBigC, "Entropy"];
highEntBigC = entropyImagesBigC[[Ordering[entropiesBigC, -10]]];
lowEntBigC = entropyImagesBigC[[Ordering[entropiesBigC, 10]]];
Thread[highEntBigC → netECA13[highEntBigC]]
Thread[lowEntBigC → netECA13[lowEntBigC]]
```

Out[•]=



Out[•]=



Network XIV - BatchNorm, 1024 linear, dropout

In[•]:= `netECA14 = netEightCC512drop[128, 128]`

Out[•]= NetChain[

		image
	Input	array (size: 3 × 128 × 128)
conv1	ConvolutionLayer	array (size: 24 × 126 × 126)
bat1	BatchNormalizationLayer	array (size: 24 × 126 × 126)
ramp1	Ramp	array (size: 24 × 126 × 126)
conv2	ConvolutionLayer	array (size: 16 × 125 × 124)
bat2	BatchNormalizationLayer	array (size: 16 × 125 × 124)
ramp2	Ramp	array (size: 16 × 125 × 124)
conv3	ConvolutionLayer	array (size: 24 × 123 × 122)
bat3	BatchNormalizationLayer	array (size: 24 × 123 × 122)
ramp3	Ramp	array (size: 24 × 123 × 122)
pooling	PoolingLayer	array (size: 24 × 4 × 11)
flatten	FlattenLayer	vector (size: 1056)
linear	LinearLayer	vector (size: 1024)
drop2	DropoutLayer	vector (size: 1024)
linear2	LinearLayer	vector (size: 4)
softmax	SoftmaxLayer	vector (size: 4)
	Output	class

]

```
netECA14 = NetTrain[netECA14, fullTrainingBigC13,
  MaxTrainingRounds → 20, BatchSize → 256 * 4, TargetDevice → "CPU",
  TrainingProgressCheckpointing → {"Directory", dir}]
```

In[•]:=
```
dir = SetDirectory[NotebookDirectory[]]
```

Out[•]=
```
/Users/thorsilver/Downloads/Wolfram notebooks
```

In[•]:=
```
netECA14 = Import["netECA14-r20.wlnet"]
```

Out[•]=
NetChain[

	Input	image array (size: 3 × 128 × 128)
conv1	ConvolutionLayer	array (size: 24 × 126 × 126)
bat1	BatchNormalizationLayer	array (size: 24 × 126 × 126)
ramp1	Ramp	array (size: 24 × 126 × 126)
conv2	ConvolutionLayer	array (size: 16 × 125 × 124)
bat2	BatchNormalizationLayer	array (size: 16 × 125 × 124)
ramp2	Ramp	array (size: 16 × 125 × 124)
conv3	ConvolutionLayer	array (size: 24 × 123 × 122)
bat3	BatchNormalizationLayer	array (size: 24 × 123 × 122)
ramp3	Ramp	array (size: 24 × 123 × 122)
pooling	PoolingLayer	array (size: 24 × 4 × 11)
flatten	FlattenLayer	vector (size: 1056)
linear	LinearLayer	vector (size: 1024)
drop2	DropoutLayer	vector (size: 1024)
linear2	LinearLayer	vector (size: 4)
softmax	SoftmaxLayer	vector (size: 4)
	Output	class

]

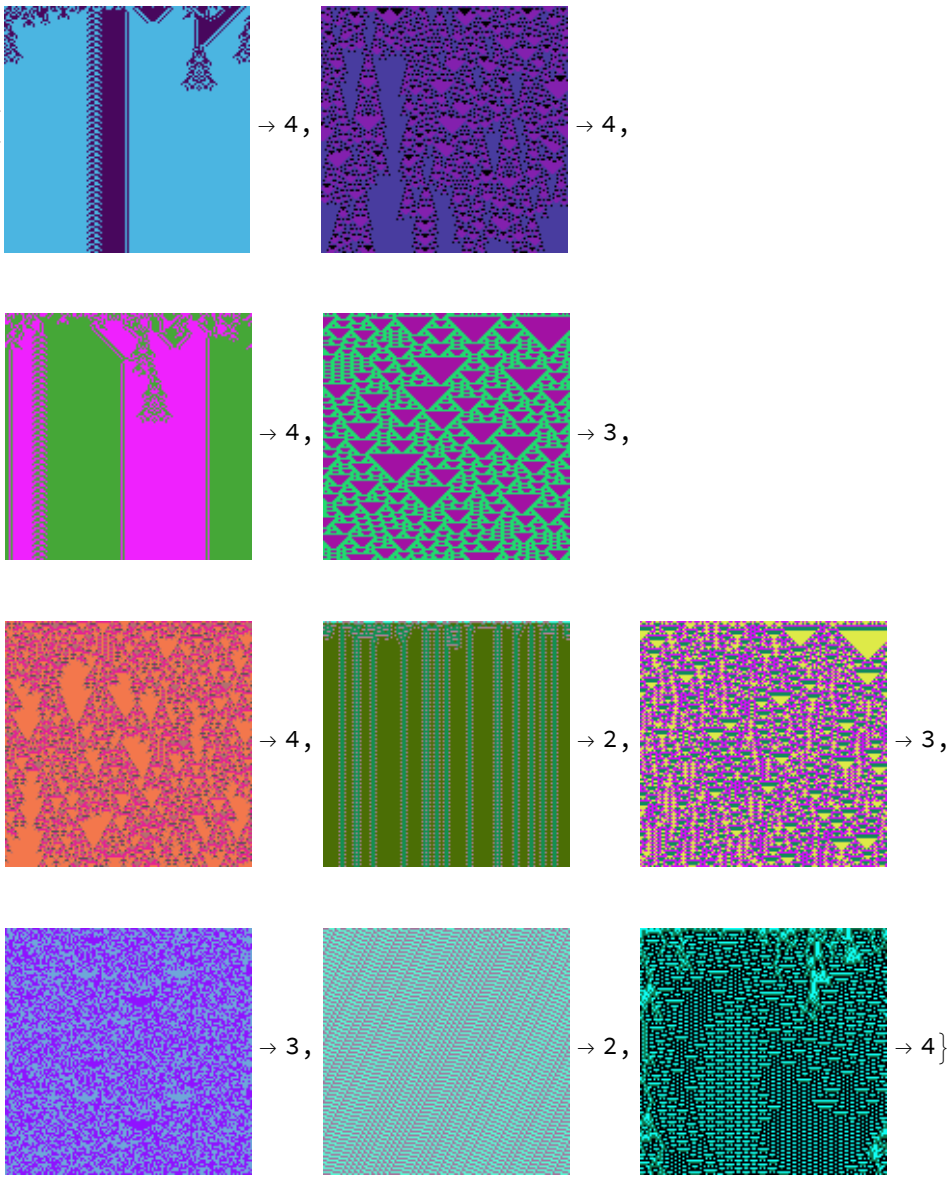
Generating test data for Network XIV

In[•]:=
```
testDataECABigC = dataC[128, 128, 1024];
testData2TBigC = genData2r2C[128, 128, 1024];
testData3TBigC = data3T2C[128, 128, 1024];
testData4TBigC = data4TC[128, 128, 1024];
testData5TBigC = genData5TCC[128, 128, 1024];
fullTestSetBigC = Join[testDataECABigC,
   testData2TBigC, testData3TBigC, testData4TBigC, testData5TBigC];
Length[fullTestSetBigC]
```

Out[•]=
```
10 240
```

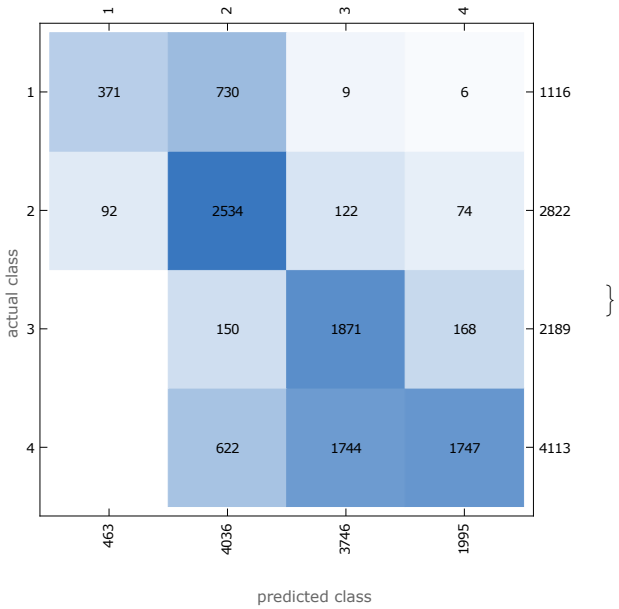
In[•]:= `RandomSample[fullTestSetBigC, 10]`

Out[•]=



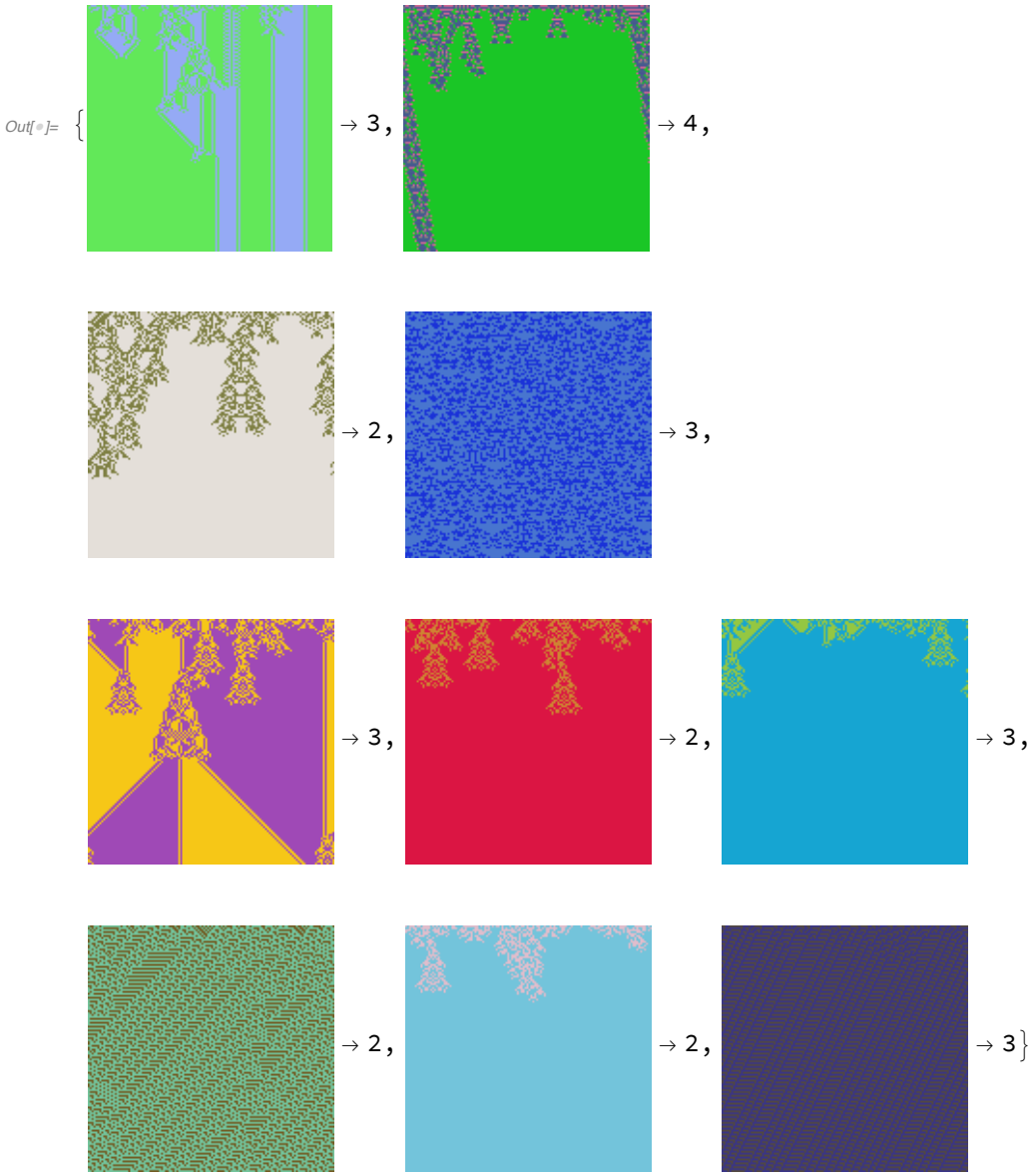
In[•]:=
```
NetMeasurements[netECA14, fullTestSetBigC,
 {"Accuracy", "Precision", "ConfusionMatrixPlot"}]
```

Out[•]=
```
{0.637012, ⟨|1 → 0.801296, 2 → 0.627849, 3 → 0.499466, 4 → 0.875689|⟩,
```

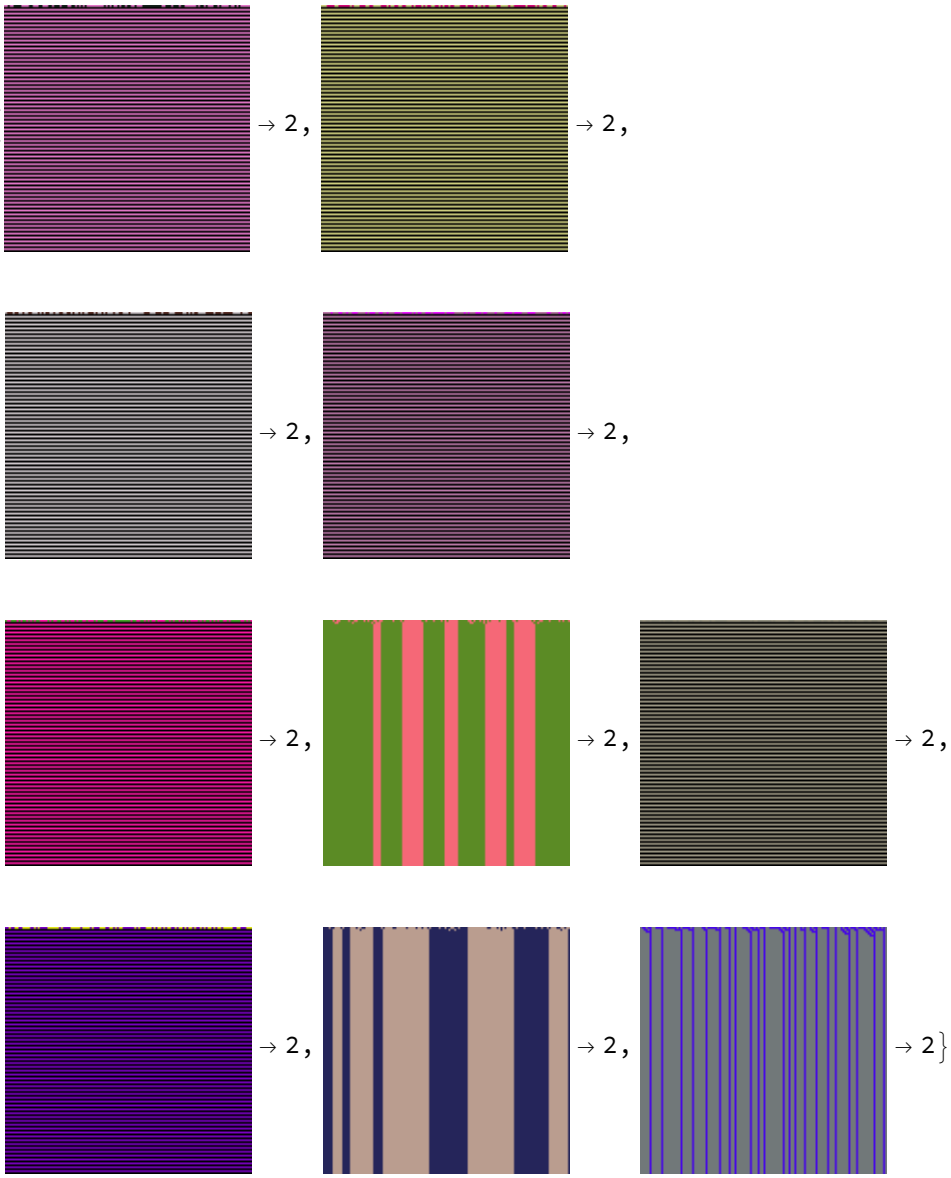



```
}
```

In[•]:=
```
entropyImagesBigC = RandomSample[Keys[fullTestSetBigC], 500];
entropiesBigC = netECA14[entropyImagesBigC, "Entropy"];
highEntBigC = entropyImagesBigC[[Ordering[entropiesBigC, -10]]];
lowEntBigC = entropyImagesBigC[[Ordering[entropiesBigC, 10]]];
Thread[highEntBigC → netECA14[highEntBigC]]
Thread[lowEntBigC → netECA14[lowEntBigC]]
```

Out[•]= 

Out[•]=



Network XV - Transfer learning with pre-trained image recognition net (VGG-16)

```
In[•]:= netECA15 = NetModel["VGG-16 Trained on ImageNet Competition Data"]
```

Out[•]= NetChain

	Input	image array (size: 3 × 224 × 224)
conv1_1	ConvolutionLayer	array (size: 64 × 224 × 224)
relu1_1	Ramp	array (size: 64 × 224 × 224)
conv1_2	ConvolutionLayer	array (size: 64 × 224 × 224)
relu1_2	Ramp	array (size: 64 × 224 × 224)
pool1	PoolingLayer	array (size: 64 × 112 × 112)
conv2_1	ConvolutionLayer	array (size: 128 × 112 × 112)
relu2_1	Ramp	array (size: 128 × 112 × 112)
conv2_2	ConvolutionLayer	array (size: 128 × 112 × 112)
relu2_2	Ramp	array (size: 128 × 112 × 112)
pool2	PoolingLayer	array (size: 128 × 56 × 56)
conv3_1	ConvolutionLayer	array (size: 256 × 56 × 56)
relu3_1	Ramp	array (size: 256 × 56 × 56)
conv3_2	ConvolutionLayer	array (size: 256 × 56 × 56)
relu3_2	Ramp	array (size: 256 × 56 × 56)
conv3_3	ConvolutionLayer	array (size: 256 × 56 × 56)
relu3_3	Ramp	array (size: 256 × 56 × 56)
pool3	PoolingLayer	array (size: 256 × 28 × 28)
conv4_1	ConvolutionLayer	array (size: 512 × 28 × 28)
relu4_1	Ramp	array (size: 512 × 28 × 28)
conv4_2	ConvolutionLayer	array (size: 512 × 28 × 28)
relu4_2	Ramp	array (size: 512 × 28 × 28)
conv4_3	ConvolutionLayer	array (size: 512 × 28 × 28)
relu4_3	Ramp	array (size: 512 × 28 × 28)
pool4	PoolingLayer	array (size: 512 × 14 × 14)
conv5_1	ConvolutionLayer	array (size: 512 × 14 × 14)
relu5_1	Ramp	array (size: 512 × 14 × 14)
conv5_2	ConvolutionLayer	array (size: 512 × 14 × 14)
relu5_2	Ramp	array (size: 512 × 14 × 14)
conv5_3	ConvolutionLayer	array (size: 512 × 14 × 14)
relu5_3	Ramp	array (size: 512 × 14 × 14)
pool5	PoolingLayer	array (size: 512 × 7 × 7)
flatten_0	FlattenLayer	vector (size: 25088)
fc6	LinearLayer	vector (size: 4096)
relu6	Ramp	vector (size: 4096)
drop6	DropoutLayer	vector (size: 4096)
fc7	LinearLayer	vector (size: 4096)
relu7	Ramp	vector (size: 4096)
drop7	DropoutLayer	vector (size: 4096)
fc8	LinearLayer	vector (size: 1000)
prob	SoftmaxLayer	vector (size: 1000)
	Output	class

```
In[•]:= subNet = NetTake[netECA15, {"conv1_1", "flatten_0"}]
```

Out[•]= NetChain

Input port:	image
Output port:	vector (size: 25088)
Number of layers:	32

```
In[•]:= joinedNet = NetJoin[subNet,
  NetChain@<|"linear_new" → LinearLayer[1024], "linear_out" → LinearLayer[4],
    "prob" → SoftmaxLayer[]|>, "Output" → NetDecoder[{"Class", Range[1, 4]}]]
```

Out[•]= NetChain

uninitialized

Input port:	image
Output port:	class
Number of layers:	35

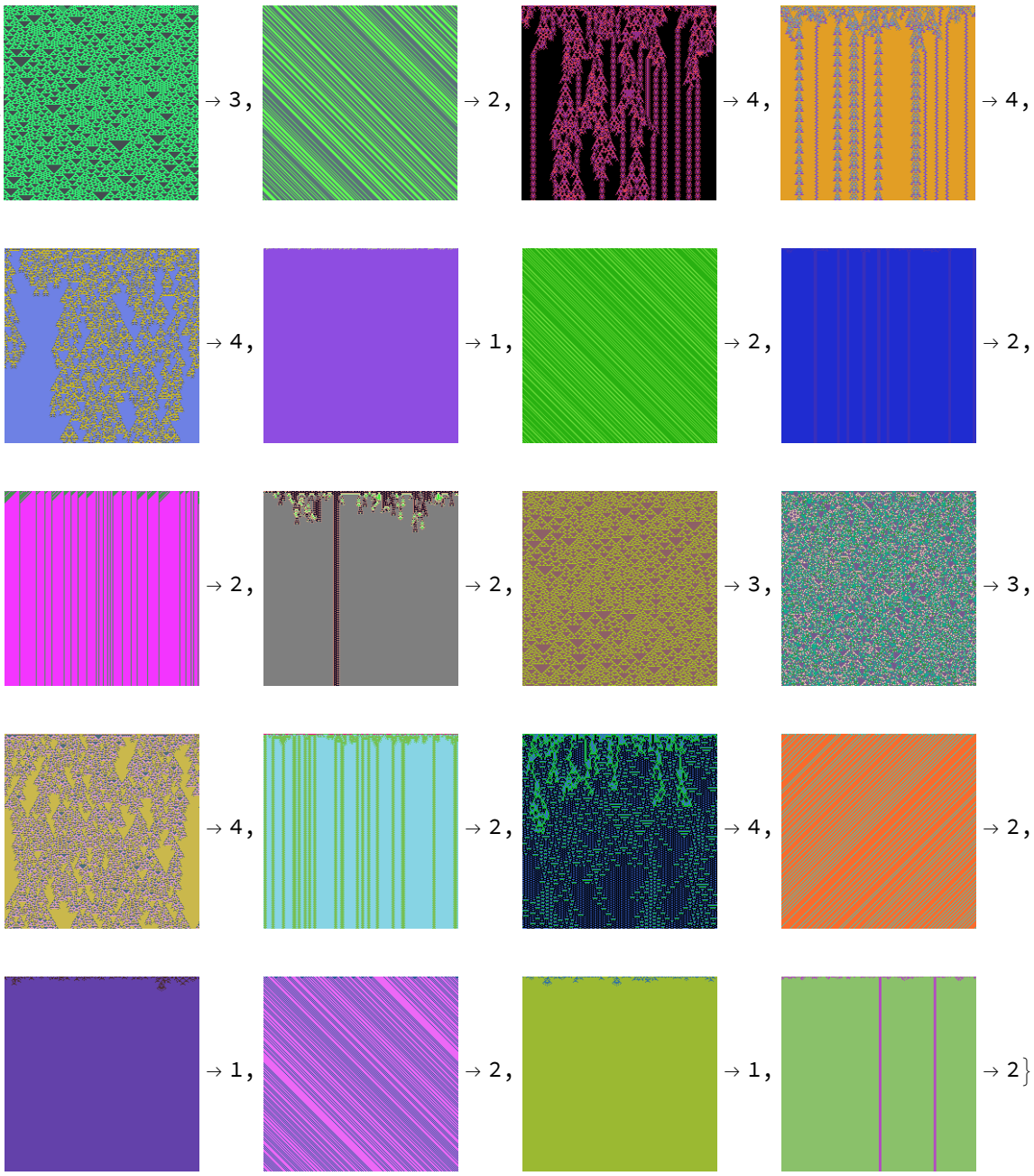
```
In[•]:= netECA15final =
         NetPrepend[joinedNet, {"augment" → ImageAugmentationLayer[{224, 224}]},
          "Input" → NetExtract[joinedNet, "Input"]]
```

Out[•]= NetChain[uninitialized | Input port: image; Output port: class; Number of layers: 36]

```
In[•]:= dataECA15 = dataC[224, 224, 8192];

In[•]:= dataTotalistic2BigC15 = genData2r2C[224, 224, 1024];

In[•]:= dataTotalistic3BigC15 = data3T2C[224, 224, 512];

In[•]:= dataTotalistic4BigC15 = data4TC[224, 224, 512];

In[•]:= dataTotalistic5BigC15 = genData5TCC[224, 224, 1024];

In[•]:= fullTrainingBigC15 = Join[dataECA15, dataTotalistic2BigC15,
           dataTotalistic3BigC15, dataTotalistic4BigC15, dataTotalistic5BigC15];
        Length[fullTrainingBigC15]
Out[•]= 16 384
```

In[•]:=
```
RandomSample[fullTrainingBigC15, 20]
```

Out[•]=



In[•]:=
```
netECA15final = NetTrain[netECA15final, fullTrainingBigC15,
  MaxTrainingRounds → 5, BatchSize → 256 * 4, TargetDevice → "CPU",
  TrainingProgressCheckpointing → {"Directory", dir},
  LearningRateMultipliers → {"linear_new" → 1, "linear_out" → 1, _ → 0}]
```

Network XVI - Three convolutions, dropout on linear only, BatchNorm

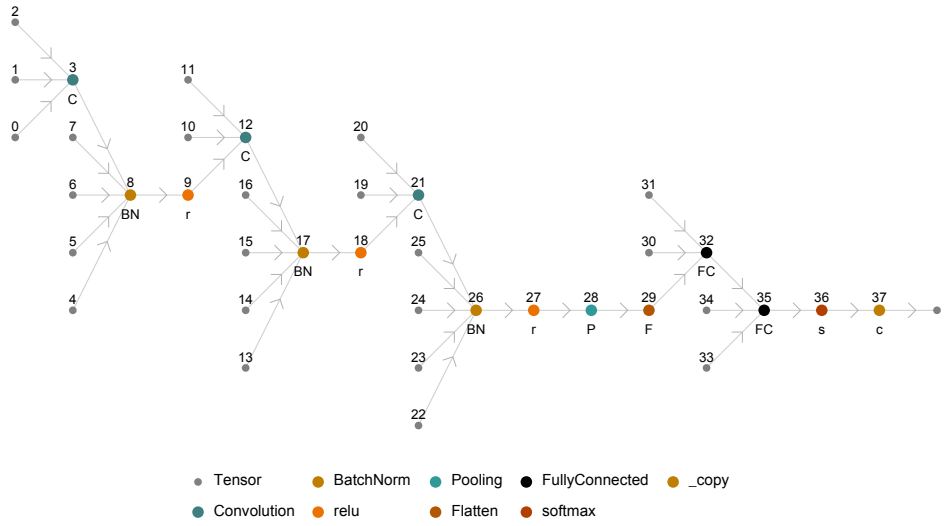
In[•]:= `netECA16 = netNineCC512drop[128, 128]`

Out[•]= NetChain

	Input	image
conv1	ConvolutionLayer	array (size: 3 × 128 × 128)
bat1	BatchNormalizationLayer	array (size: 24 × 126 × 126)
ramp1	Ramp	array (size: 24 × 126 × 126)
conv2	ConvolutionLayer	array (size: 24 × 124 × 124)
bat2	BatchNormalizationLayer	array (size: 24 × 124 × 124)
ramp2	Ramp	array (size: 24 × 124 × 124)
conv3	ConvolutionLayer	array (size: 24 × 122 × 122)
bat3	BatchNormalizationLayer	array (size: 24 × 122 × 122)
ramp3	Ramp	array (size: 24 × 122 × 122)
pooling	PoolingLayer	array (size: 24 × 7 × 7)
flatten	FlattenLayer	vector (size: 1176)
linear	LinearLayer	vector (size: 512)
drop2	DropoutLayer	vector (size: 512)
linear2	LinearLayer	vector (size: 4)
softmax	SoftmaxLayer	vector (size: 4)
	Output	class

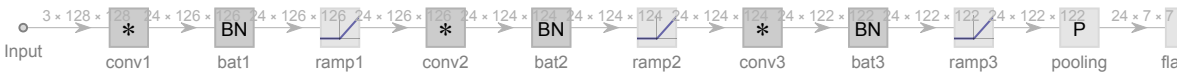
In[•]:= `NetInformation[netECA16, "MXNetNodeGraphPlot"]`

Out[•]=



In[•]:= `NetInformation[netECA16, "SummaryGraphic"]`

Out[•]=



In[•]:= `dataECA16 = dataC[128, 128, 8192];`

In[•]:= `dataTotalistic2BigC16 = genData2r2C[128, 128, 1024];`

In[•]:= `dataTotalistic3BigC16 = data3T2C[128, 128, 1024];`

In[•]:= `dataTotalistic4BigC16 = data4TC[128, 128, 1024];`

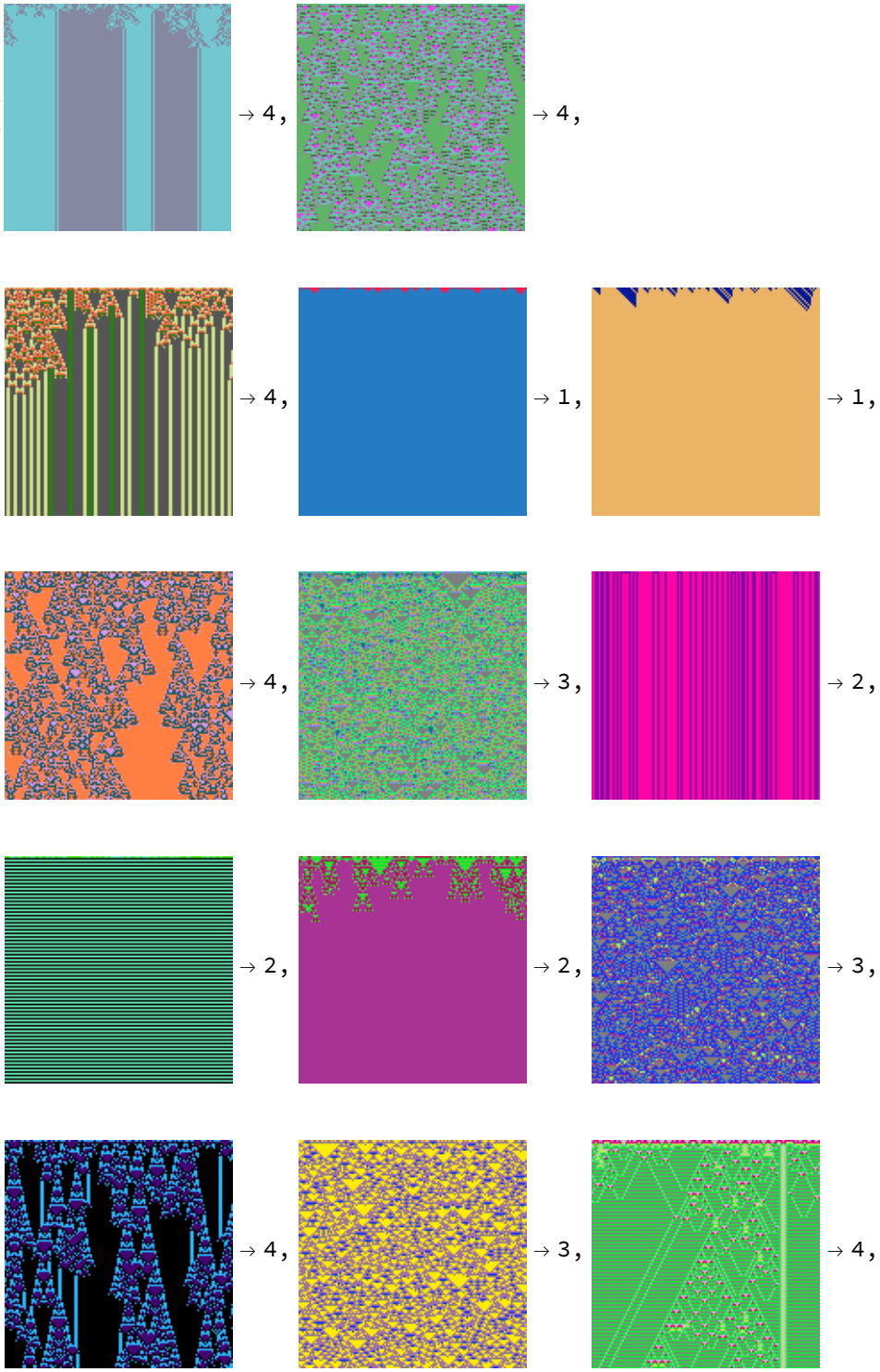
In[•]:= `dataTotalistic5BigC16 = genData5TCC[128, 128, 4096];`

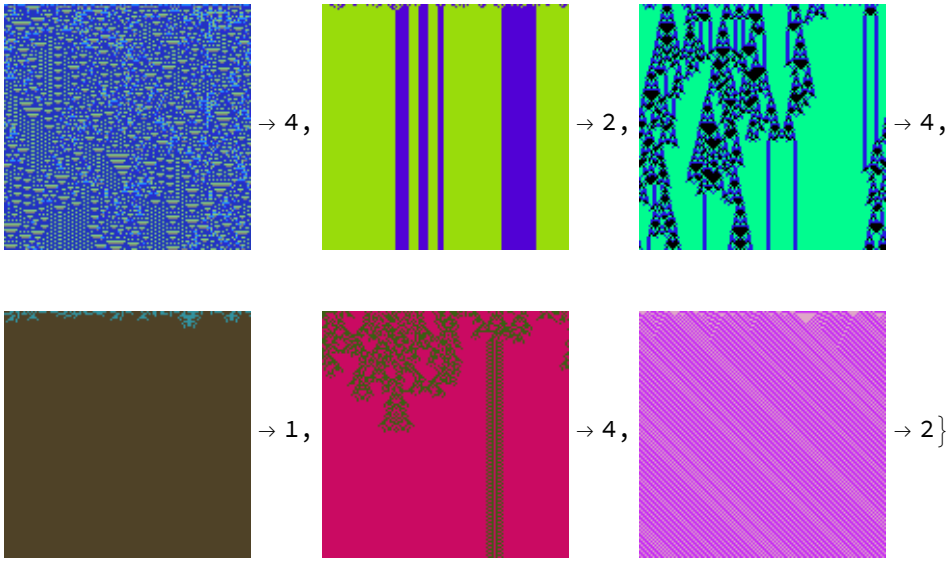
In[•]:=
```
fullTrainingBigC16 = Join[dataECA16, dataTotalistic2BigC16,
   dataTotalistic3BigC16, dataTotalistic4BigC16, dataTotalistic5BigC16];
Length[fullTrainingBigC16]
```

Out[•]= 26 624

In[•]:=
```
RandomSample[fullTrainingBigC16, 20]
```

Out[•]=





In[•]:=
```
dir = SetDirectory[NotebookDirectory[]]
```
Out[•]=
```
/Users/thorsilver/Downloads/Wolfram notebooks
```

In[•]:=
```
netECA16 = NetTrain[netECA16, fullTrainingBigC16,
  MaxTrainingRounds → 20, BatchSize → 256, TargetDevice → "CPU",
  TrainingProgressCheckpointing → {"Directory", dir}]
```

Out[•]= NetChain[Input port: image; Output port: class; Number of layers: 15]

```
netECA16 = Import["netECA16-r20.wlnet"]
```

```
netECA16 = NetTrain[netECA16, fullTrainingBigC16,
  MaxTrainingRounds → 20, BatchSize → 256 * 4, TargetDevice → "CPU",
  TrainingProgressCheckpointing → {"Directory", dir}]
```

Generate test data for Network XVI

In[57]:=
```
dir = SetDirectory[NotebookDirectory[]]
```
Out[57]=
```
/Users/thorsilver/Downloads/Wolfram notebooks
```

In[58]:=
```
netECA16 = Import["netECA16-r20.wlnet"]
```

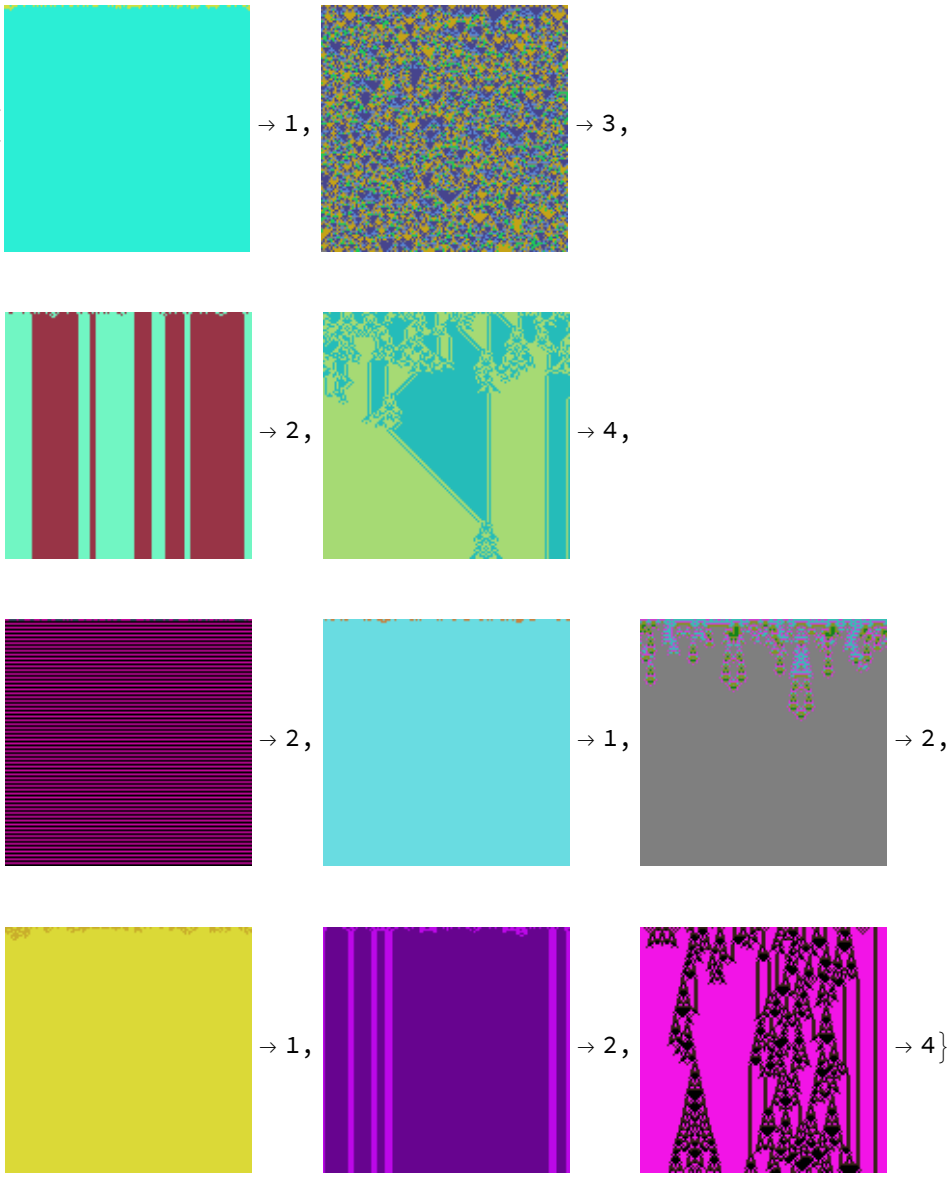
Out[58]= NetChain[Input port: image; Output port: class; Number of layers: 15]

In[•]:=
```
testDataECABigC = dataC[128, 128, 1024];
testData2TBigC = genData2r2C[128, 128, 1024];
testData3TBigC = data3T2C[128, 128, 1024];
testData4TBigC = data4TC[128, 128, 1024];
testData5TBigC = genData5TCC[128, 128, 1024];
fullTestSetBigC = Join[testDataECABigC,
   testData2TBigC, testData3TBigC, testData4TBigC, testData5TBigC];
Length[fullTestSetBigC]
```

Out[•]=
```
10 240
```

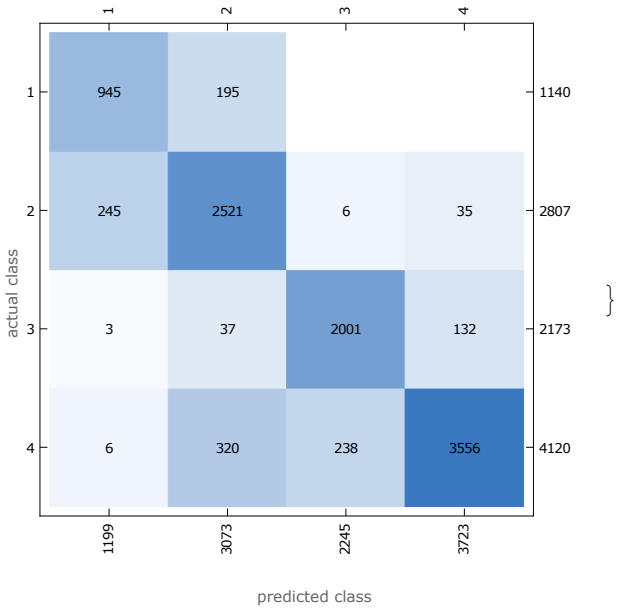
In[•]:=
```
RandomSample[fullTestSetBigC, 10]
```

Out[•]=



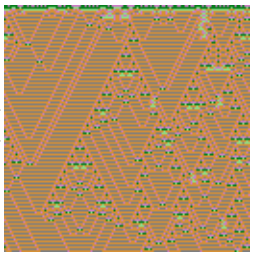
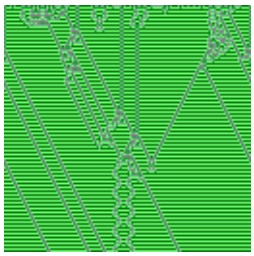
In[•]:=
```
NetMeasurements[netECA16, fullTestSetBigC,
 {"Accuracy", "Precision", "ConfusionMatrixPlot"}]
```

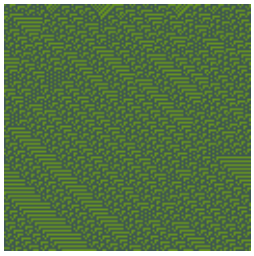
Out[•]=
```
{0.881152, ⟨|1 → 0.788157, 2 → 0.820371, 3 → 0.891314, 4 → 0.955144|⟩,
```

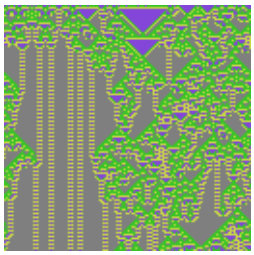
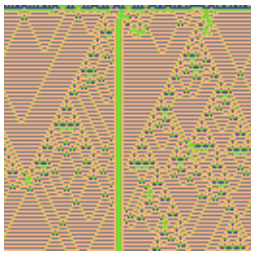


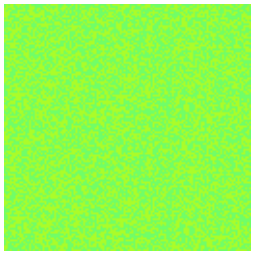

```
}
```

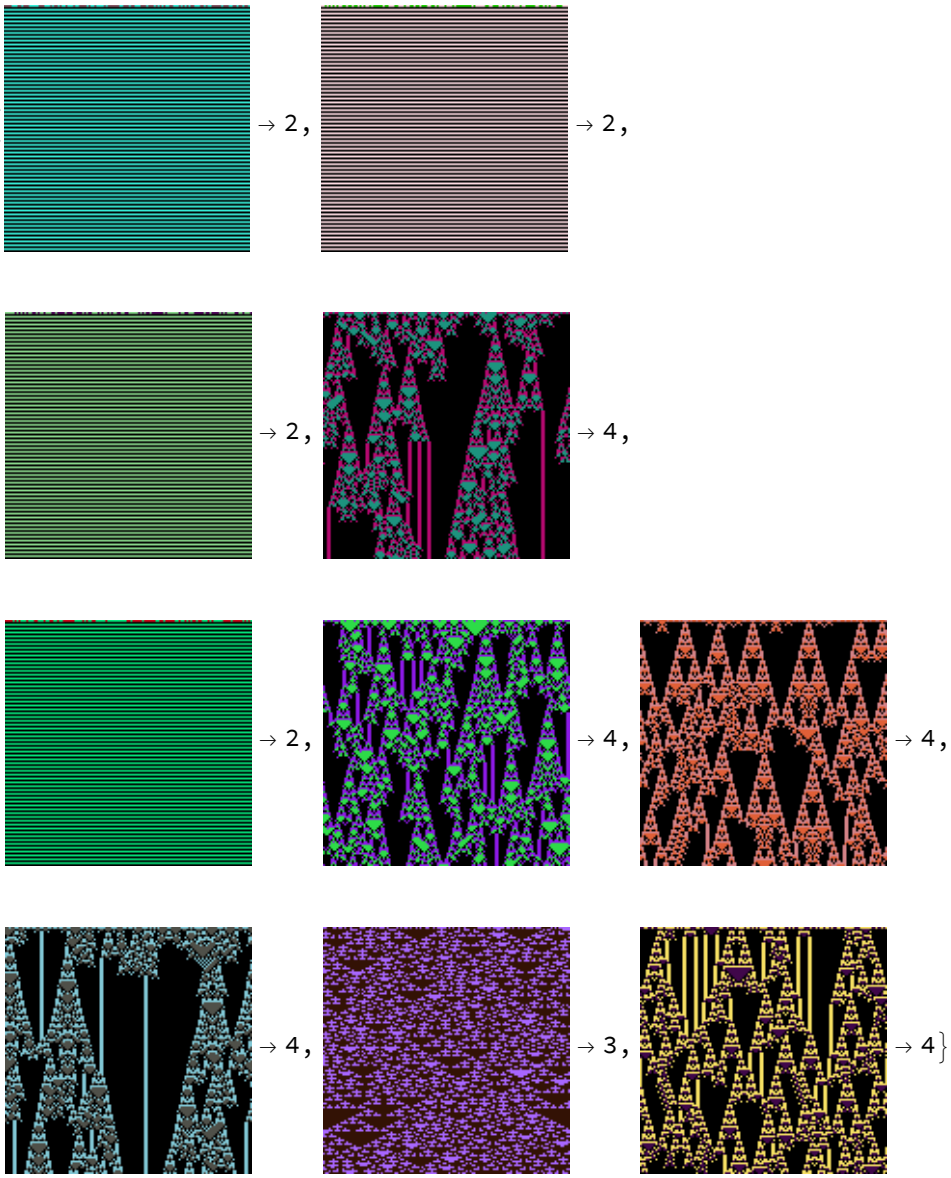
In[•]:=
```
entropyImagesBigC = RandomSample[Keys[fullTestSetBigC], 500];
entropiesBigC = netECA16[entropyImagesBigC, "Entropy"];
highEntBigC = entropyImagesBigC[[Ordering[entropiesBigC, -10]]];
lowEntBigC = entropyImagesBigC[[Ordering[entropiesBigC, 10]]];
Thread[highEntBigC → netECA16[highEntBigC]]
Thread[lowEntBigC → netECA16[lowEntBigC]]
```

Out[•]= {  → 4,  → 4,

 → 4,  → 3,

 → 4,  → 3,  → 2,

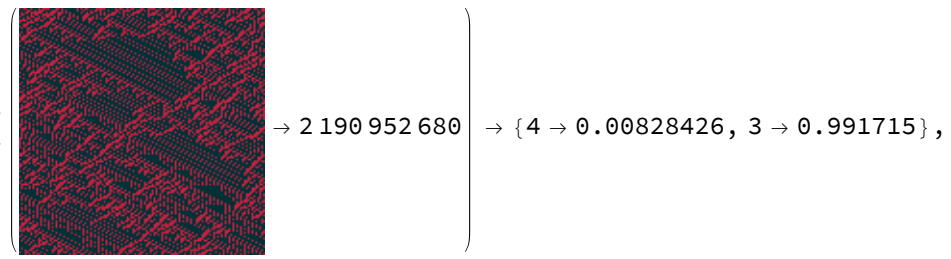
 → 4,  → 2,  → 2 }

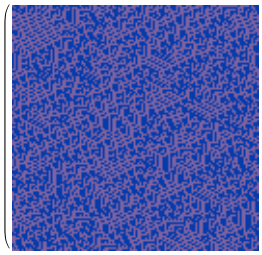
Out[•]= 

Testing Network XVI on unseen CA rule spaces

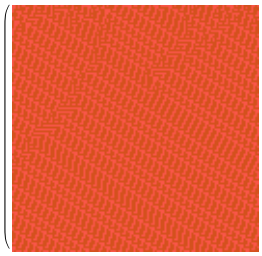
2-colour non-totalistic, range 2

In[•]:=
```
test4Data2kr2C16 = datak2r2C[128, 128, 8];
Thread[
 test4Data2kr2C16 → netECA16[Keys@test4Data2kr2C16, {"TopProbabilities", 2}]]
```

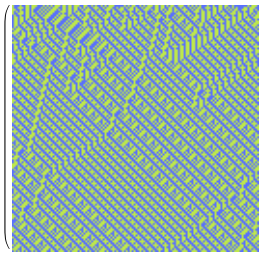
Out[•]= 



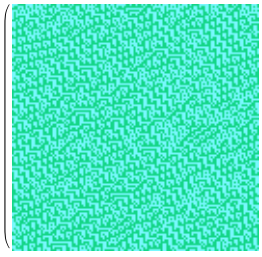
→ 452 760 995 → {4 → 0.0697559, 3 → 0.930224},



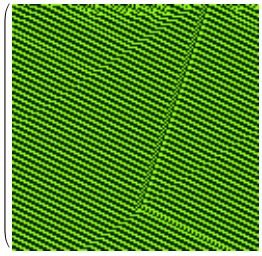
→ 1 629 569 269 → {1 → 0.0640571, 2 → 0.933385},



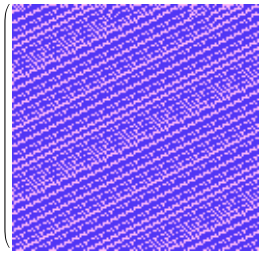
→ 2 240 312 508 → {3 → 0.118588, 2 → 0.799924},



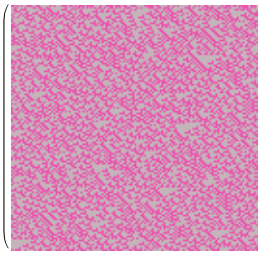
→ 482 556 537 → {4 → 0.0846837, 3 → 0.914806},



→ 1 597 446 147 → {4 → 0.248082, 2 → 0.566962},



→ 2 592 600 046 → {3 → 0.111319, 2 → 0.801882},


→ 2 548 551 192 → {4 → 0.0691249, 3 → 0.930844}}

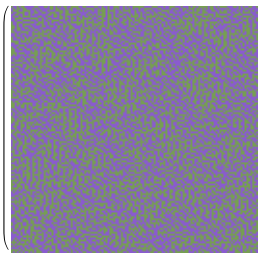
2-colour non-totalistic, range 3

In[81]:=
```
test4Data2kr3C16 = datak2r3NT[128, 128, 8];
Thread[
 test4Data2kr3C16 → netECA16[Keys@test4Data2kr3C16, {"TopProbabilities", 2}]]
```

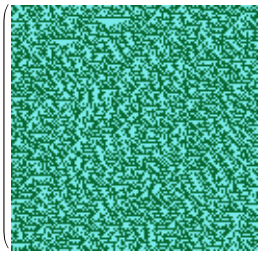
Out[82]=


→ 192 512 040 898 603 851 305 563 821 357 837 920 322 →

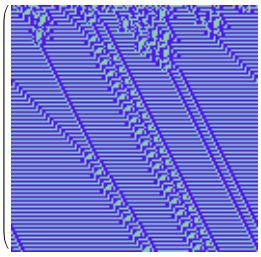
{3 → 0.277165, 4 → 0.649887},


→ 268 814 102 680 703 659 513 703 900 569 271 732 433 →

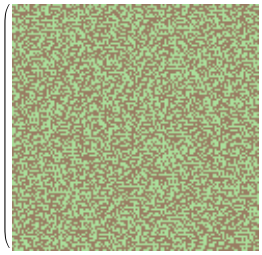
{4 → 0.410444, 3 → 0.585722},


→ 234 179 531 560 333 249 757 404 371 202 280 747 219 →

{4 → 0.223487, 3 → 0.776512},


→ 25 383 678 045 436 598 053 387 497 115 518 594 687 →

{3 → 0.0605962, 4 → 0.938855},



→ 198 476 619 696 798 272 452 027 738 615 092 731 693 →

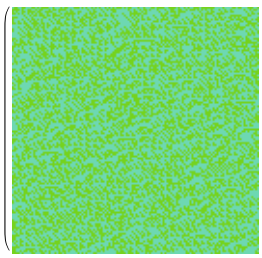
{4 → 0.0598654, 3 → 0.940094},

→ 338 073 225 958 790 856 870 507 077 934 439 630 490 →

{4 → 0.354784, 2 → 0.643357},

→ 327 367 588 377 016 075 437 244 034 595 798 938 936 →

{3 → 0.0133918, 4 → 0.986511},



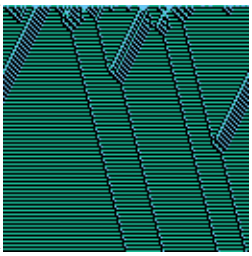
→ 64 210 583 158 946 454 226 752 791 209 536 621 761 →

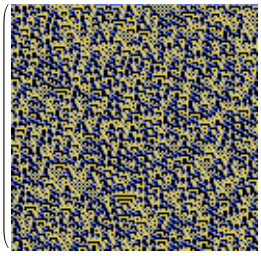
{3 → 0.327669, 4 → 0.672036}}

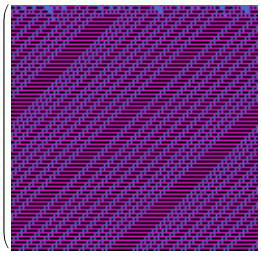
3-colour non-totalistic, range 1

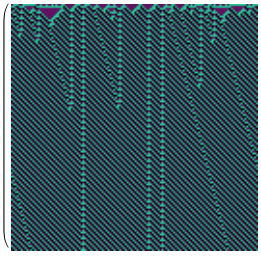
In[89]:=
```
test4Data3kr1C16 = datak3r1NT[128, 128, 8];
Thread[
 test4Data3kr1C16 → netECA16[Keys@test4Data3kr1C16, {"TopProbabilities", 2}]]
```

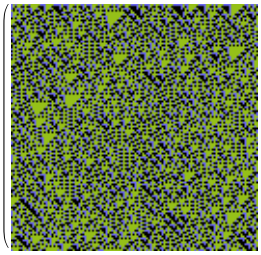
Out[90]= {

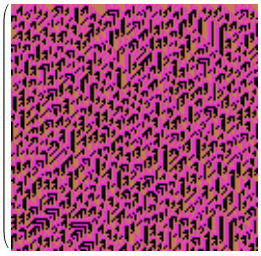

→ 2 473 850 236 436 → {4 → 0.00291686, 2 → 0.996637},


→ 2 347 219 495 748 → {4 → 0.158002, 3 → 0.841998},

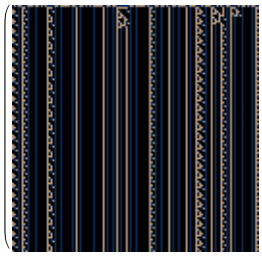

→ 851 165 089 277 → {4 → 0.423993, 2 → 0.575989},


→ 6 208 032 017 586 → {3 → 0.291377, 2 → 0.418742},


→ 7 322 710 481 736 → {3 → 0.0417726, 4 → 0.95685},


→ 4 864 402 493 091 → {4 → 0.075809, 3 → 0.924191},

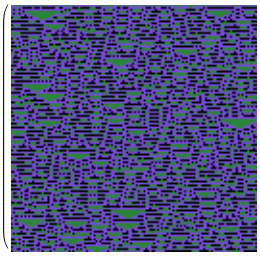

→ 4 457 586 281 952 → {1 → 0.000285388, 2 → 0.999711},

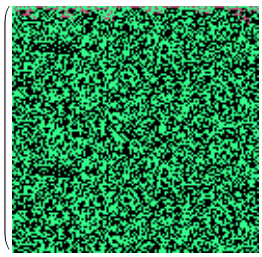

→ 7 458 524 800 199 → {1 → 0.00964059, 2 → 0.990356}}

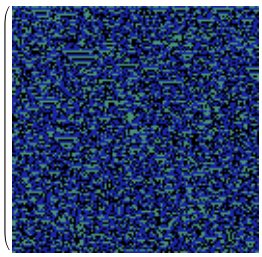
3-colour totalistic, range 2

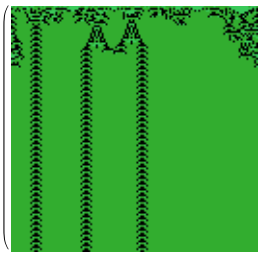
In[•]:=
```
test4Data3kr2C16 = datak3r2C[128, 128, 8];
Thread[
 test4Data3kr2C16 → netECA16[Keys@test4Data3kr2C16, {"TopProbabilities", 2}]]
```

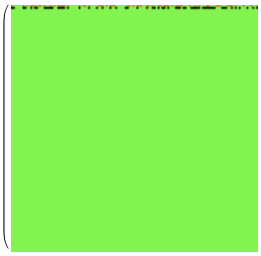
Out[•]=

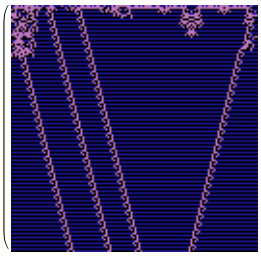

→ 22 138 → {3 → 0.432672, 4 → 0.567328},

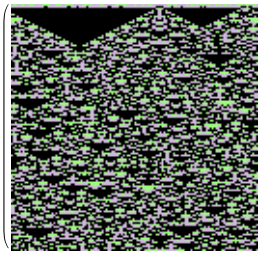

→ 15 962 → {4 → 0.0786194, 3 → 0.921381},

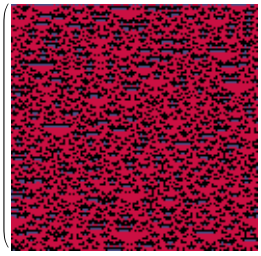

→ 14 359 → {3 → 0.285852, 4 → 0.713315},


→ 84 285 → {3 → 0.0456504, 4 → 0.953803},


→ 88 604 → {2 → 0.0365346, 1 → 0.963464},


→ 105 626 → {1 → 0.00134017, 2 → 0.998622},


→ 131 100 → {4 → 0.126999, 3 → 0.873001},

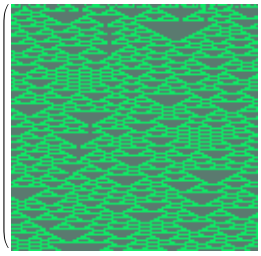

→ 74 382 → {4 → 0.00764391, 3 → 0.992356}}

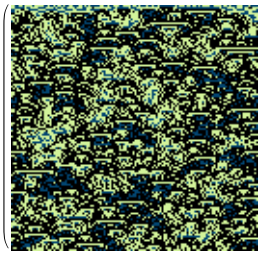
3-colour totalistic, range 3

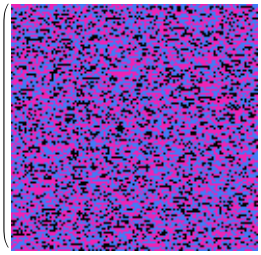
In[•]:=
```
test4Data3kr3C16 = datak3r3C[128, 128, 8];
Thread[
 test4Data3kr3C16 → netECA16[Keys@test4Data3kr3C16, {"TopProbabilities", 2}]]
```

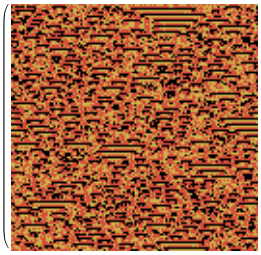
Out[•]= {

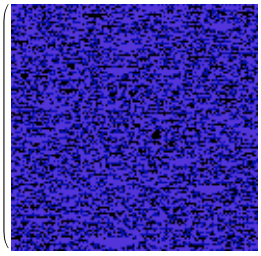
 → 13 965 620 → {4 → 0.129201, 2 → 0.823269},

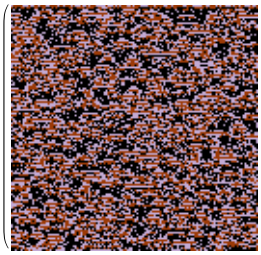
 → 2 834 469 → {4 → 0.00281357, 3 → 0.997186},

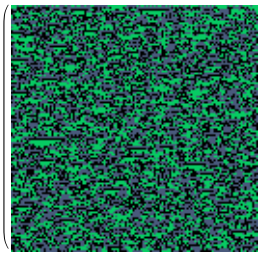
 → 4 592 009 → {3 → 0.13601, 4 → 0.86399},

 → 5 085 210 → {3 → 0.0479753, 4 → 0.952025},

 → 5 151 167 → {3 → 0.491086, 4 → 0.508913},

 → 12 114 410 → {4 → 0.230657, 3 → 0.769343},

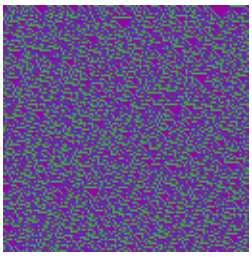

→ 2 816 030 → {3 → 0.0175618, 4 → 0.982438},


→ 5 685 248 → {4 → 0.169957, 3 → 0.830043}}

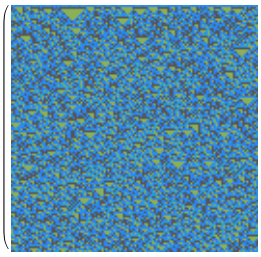
4-colour non-totalistic, range 1

In[93]:=
```
test4Data4kr1C16 = datak4r1NT[128, 128, 8];
Thread[
 test4Data4kr1C16 → netECA16[Keys@test4Data4kr1C16, {"TopProbabilities", 2}]]
```

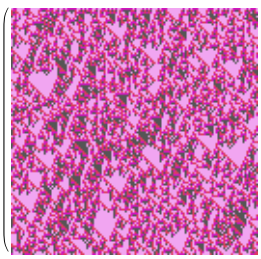
Out[94]=


→ 24 580 558 256 129 418 488 831 337 020 120 596 557 →

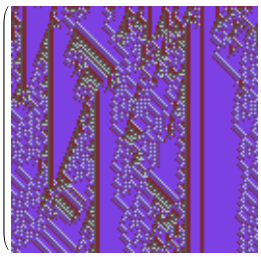
{4 → 0.0591881, 3 → 0.940812},


→ 320 024 124 365 477 690 559 134 299 487 833 253 915 →

{4 → 0.0960659, 3 → 0.903922},

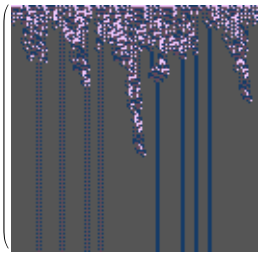

→ 106 948 342 425 382 488 412 301 983 608 519 263 756 →

{4 → 0.112148, 3 → 0.887845},



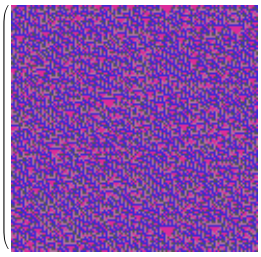
→ 86 351 110 305 322 460 128 720 263 758 500 775 714 →

{4 → 0.210001, 2 → 0.788983},



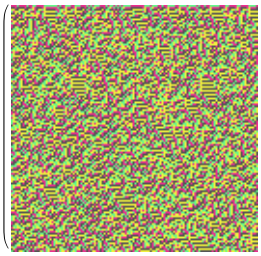
→ 77 433 156 501 235 157 602 475 680 472 280 279 759 →

{2 → 0.474925, 4 → 0.524655},



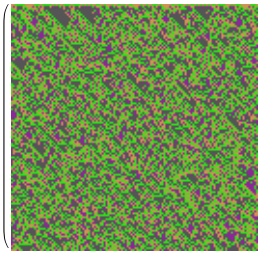
→ 596 023 145 536 016 638 713 432 023 142 370 353 →

{4 → 0.0241829, 3 → 0.975817},



→ 185 023 579 802 380 747 817 653 061 728 311 135 906 →

{4 → 0.0751294, 3 → 0.924824},



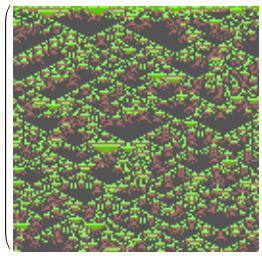
→ 273 295 602 940 247 217 835 514 841 293 393 314 396 →

{4 → 0.201624, 3 → 0.797811}}

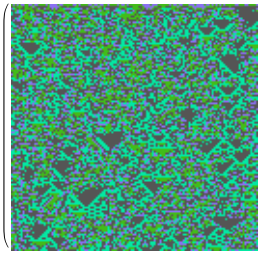
4-colour totalistic, range 2

In[•]:=
```
test4Data4kr2C16 = datak4r2C[128, 128, 8];
Thread[
 test4Data4kr2C16 → netECA16[Keys@test4Data4kr2C16, {"TopProbabilities", 2}]]
```

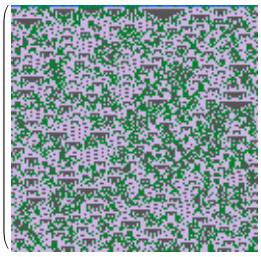
Out[•]=



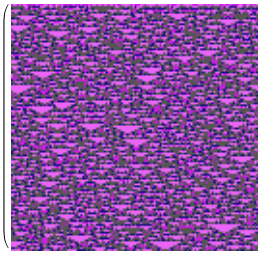
→ 1 806 102 772 → {4 → 0.0919674, 3 → 0.908033},



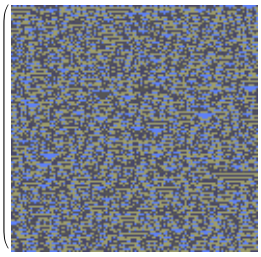
→ 1 872 979 601 → {4 → 0.349808, 3 → 0.650192},



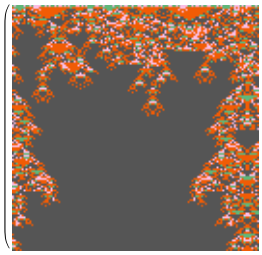
→ 3 186 088 319 → {4 → 0.0374994, 3 → 0.962501},

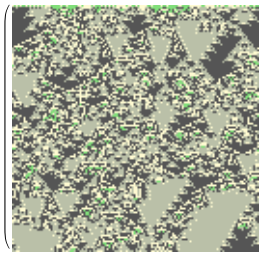


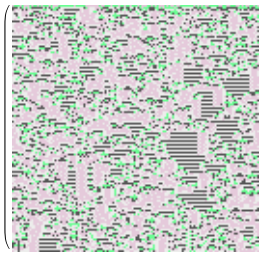
→ 271 280 936 → {4 → 0.000901956, 3 → 0.999098},



→ 1 898 315 512 → {4 → 0.00981637, 3 → 0.990184},


→ 3 684 955 635 → {3 → 0.190496, 4 → 0.809504},

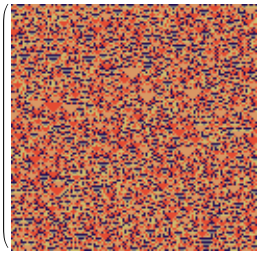

→ 3 752 298 976 → {4 → 0.0541872, 3 → 0.945813},

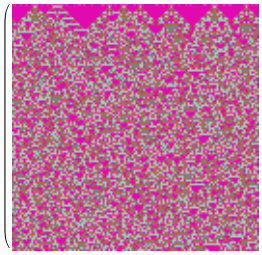

→ 2 566 352 175 → {3 → 0.0379958, 4 → 0.962004}}

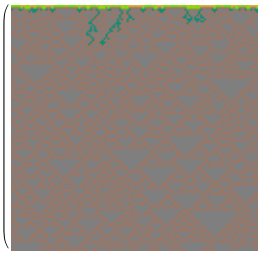
5-colour totalistic, range 1

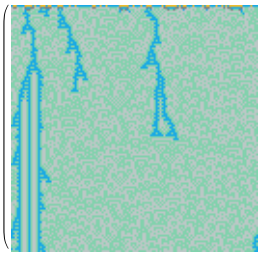
In[•]:=
```
test4Data5kr1C16 = data5T2C[8, 128, 128];
Thread[
 test4Data5kr1C16 → netECA16[Keys@test4Data5kr1C16, {"TopProbabilities", 2}]]
```

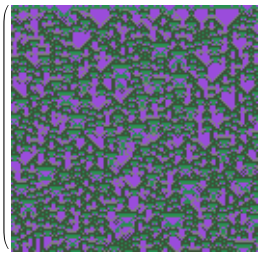
Out[•]=

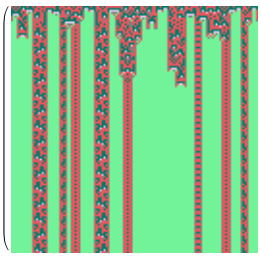

→ 407 892 722 → {4 → 0.445788, 3 → 0.554177},

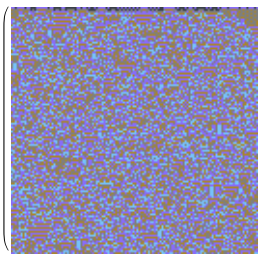

→ 638 248 908 → {4 → 0.0228094, 3 → 0.97719},

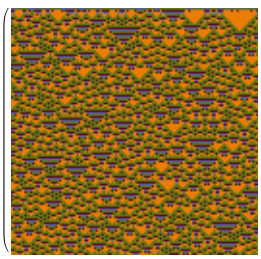

→ 710 896 859 → {1 → 0.00682515, 2 → 0.987484},


→ 966 655 073 → {3 → 0.161564, 4 → 0.837494},


→ 411 613 899 → {4 → 0.0818569, 3 → 0.918143},


→ 554 163 154 → {1 → 0.00503217, 2 → 0.99461},

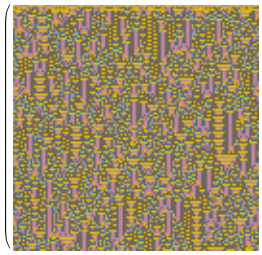

→ 676 388 552 → {4 → 0.300972, 3 → 0.698657},

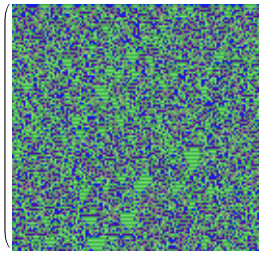

→ 134 083 721 → {4 → 0.129426, 3 → 0.870558}}

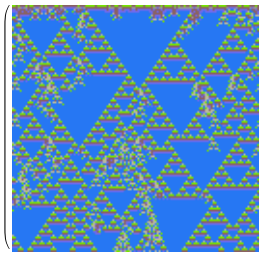
6-colour totalistic, range 1

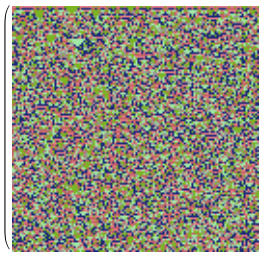
In[•]:=
```
test4Data6kr1C16 = data6TC[8, 128, 128];
Thread[
 test4Data6kr1C16 → netECA16[Keys@test4Data6kr1C16, {"TopProbabilities", 2}]]
```

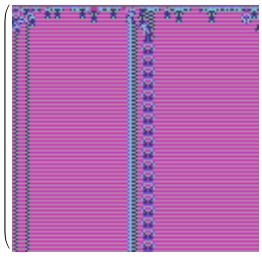
Out[•]=

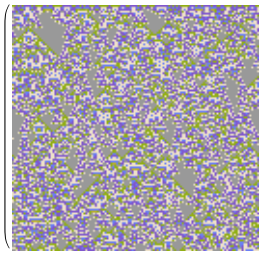

→ 930 044 739 883 → {4 → 0.0671154, 3 → 0.932882},

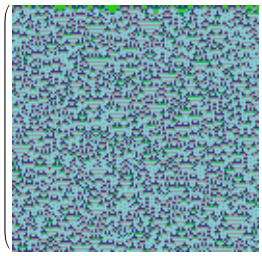

→ 1 609 022 451 969 → {3 → 0.328033, 4 → 0.671644},

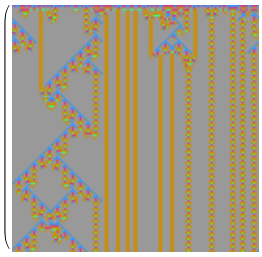

→ 2 498 882 479 071 → {3 → 0.00867874, 4 → 0.991315},


→ 250 248 309 401 → {4 → 0.0365648, 3 → 0.963435},


→ 2 382 182 512 148 → {1 → 0.0000910832, 2 → 0.999907},


→ 1 552 150 347 166 → {3 → 0.119638, 4 → 0.880047},

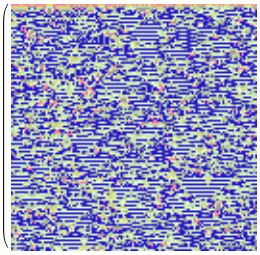

→ 72 403 648 176 → {4 → 0.0055479, 3 → 0.994452},

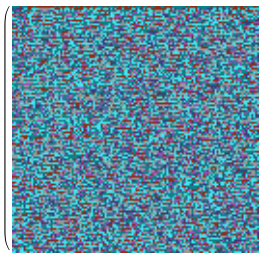

→ 621 012 387 361 → {2 → 0.284038, 4 → 0.713633}}

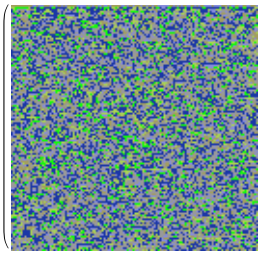
6-colour totalistic, range 2

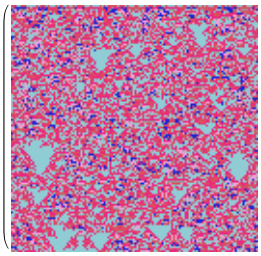
In[•]:=
```
test4Data6kr2C16 = data6T2C[8, 128, 128];
Thread[
 test4Data6kr2C16 → netECA16[Keys@test4Data6kr2C16, {"TopProbabilities", 2}]]
```

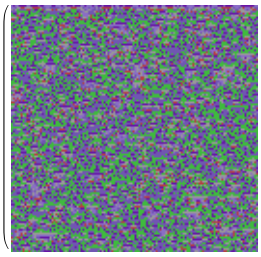
Out[•]=

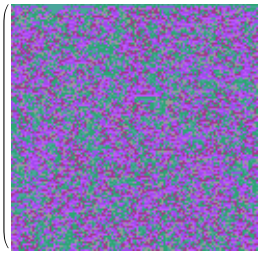

→ 68 918 261 516 486 585 431 → {3 → 0.389806, 4 → 0.607333},

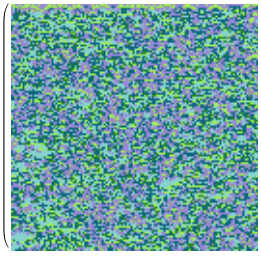

→ 81 547 637 786 331 552 954 → {3 → 0.380046, 4 → 0.619953},

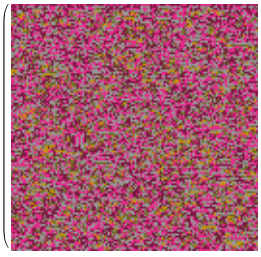

→ 27 230 916 895 777 366 626 → {4 → 0.111816, 3 → 0.888135},


→ 88 349 935 939 278 240 116 → {3 → 0.0195468, 4 → 0.980452},


→ 106 204 901 720 869 169 245 → {4 → 0.0393044, 3 → 0.960696},

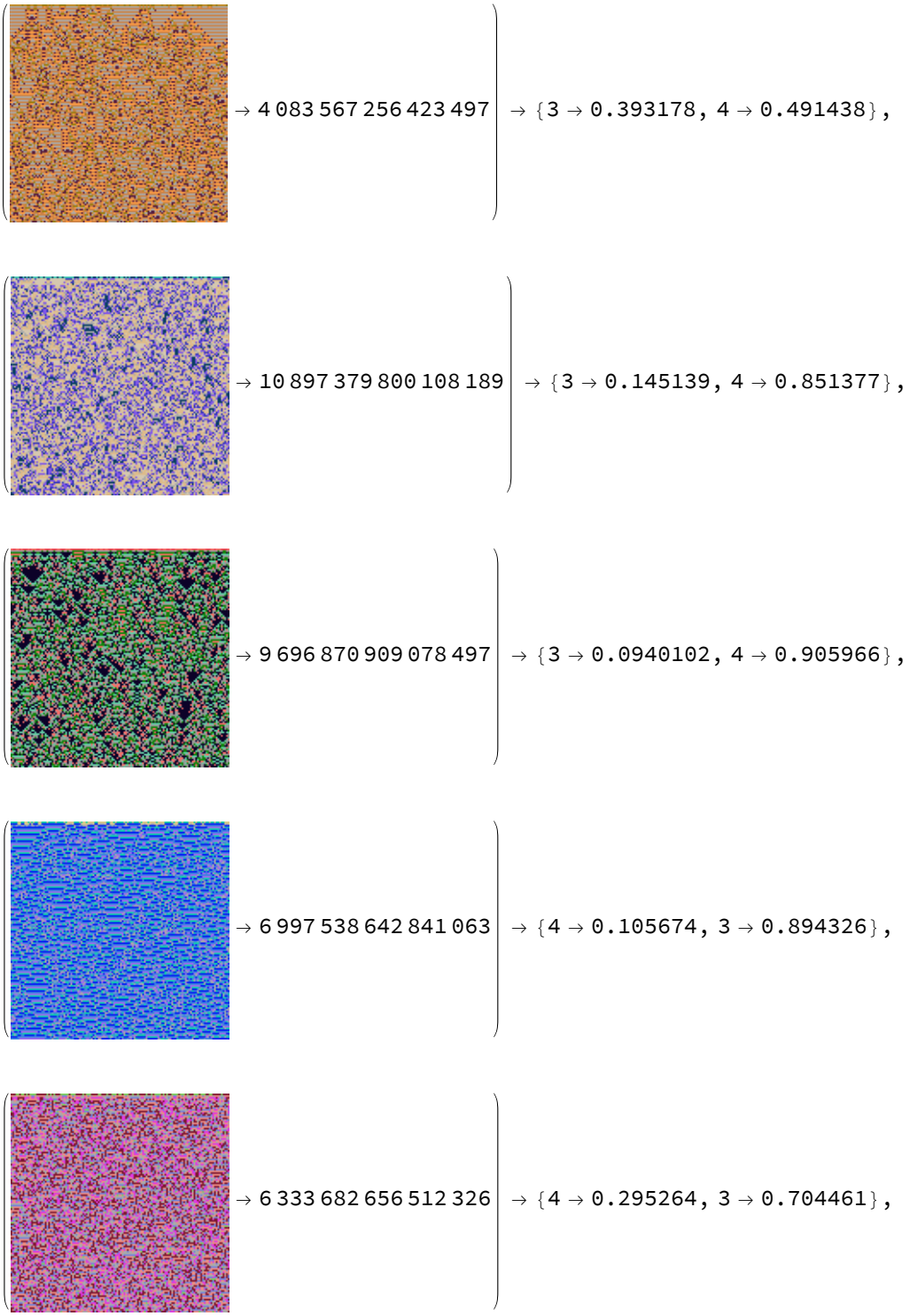

→ 12 255 387 586 435 585 434 → {4 → 0.0196859, 3 → 0.980314},

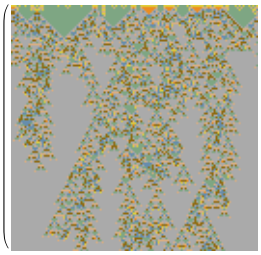

→ 52 226 526 820 494 761 535 → {4 → 0.106148, 3 → 0.893852},

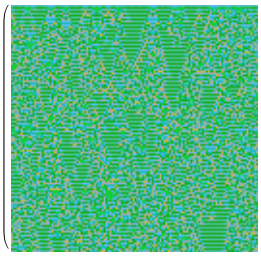

→ 62 760 542 193 998 695 205 → {4 → 0.333915, 3 → 0.666084}}

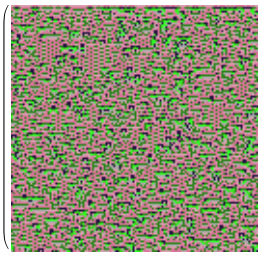
7-colour totalistic, range 1

In[•]:=
```
test4Data7kr1C16 = data7TC[8, 128, 128];
Thread[
 test4Data7kr1C16 → netECA16[Keys@test4Data7kr1C16, {"TopProbabilities", 2}]]
```

Out[•]=


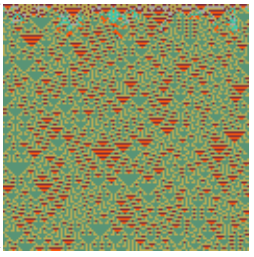

→ 4 047 667 631 402 393 → {3 → 0.292137, 4 → 0.707841},

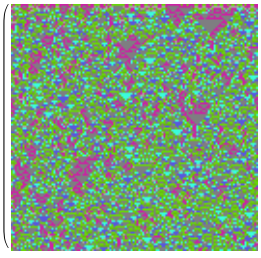

→ 10 474 570 668 116 194 → {4 → 0.291514, 3 → 0.708433},

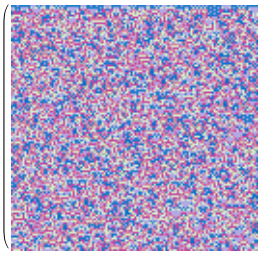

→ 8 413 297 417 285 155 → {4 → 0.269519, 3 → 0.730477}}

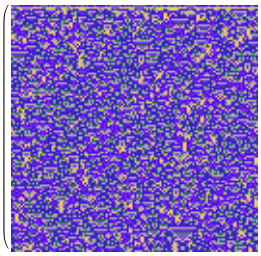
In[•]:=
```
test4Data7kr1C16 = data7TC[8, 128, 128];
Thread[
 test4Data7kr1C16 → netECA16[Keys@test4Data7kr1C16, {"TopProbabilities", 2}]]
```

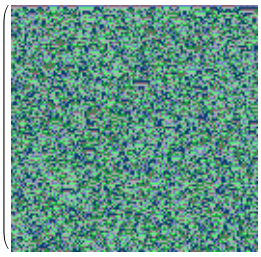
Out[•]= {

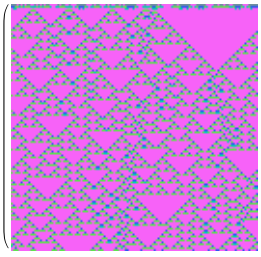

→ 11 101 279 181 647 317 → {4 → 0.250155, 3 → 0.749527},

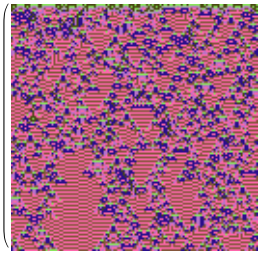

→ 11 021 473 369 162 315 → {4 → 0.0524406, 3 → 0.947559},

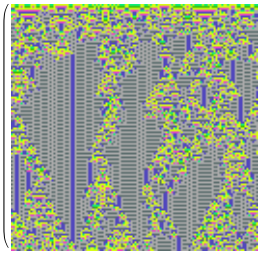

→ 4 867 969 352 368 900 → {3 → 0.253496, 4 → 0.746471},


→ 1 278 729 909 433 340 → {3 → 0.12299, 4 → 0.876993},


→ 3 704 845 844 743 256 → {4 → 0.466395, 3 → 0.533503},

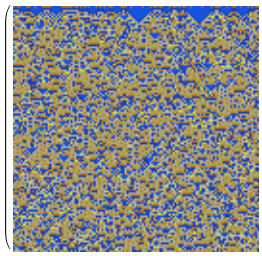

→ 4 223 308 668 057 052 → {4 → 0.488556, 3 → 0.511443},


→ 6 902 850 650 536 429 → {3 → 0.215413, 4 → 0.784586},

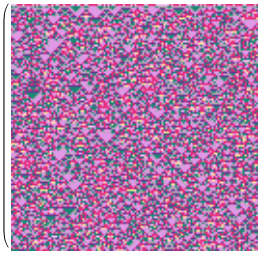

→ 1 926 864 005 427 175 → {3 → 0.0946958, 4 → 0.904136}}

In[•]:=
```
test4Data7kr1C16 = data7TC[8, 128, 128];
Thread[
 test4Data7kr1C16 → netECA16[Keys@test4Data7kr1C16, {"TopProbabilities", 2}]]
```

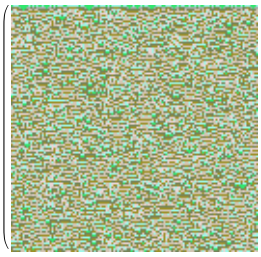
Out[•]=



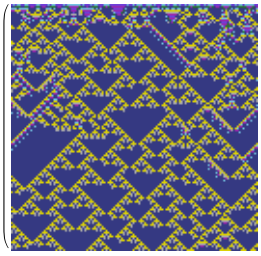
→ 203 595 605 949 565 → {4 → 0.190292, 3 → 0.809707},



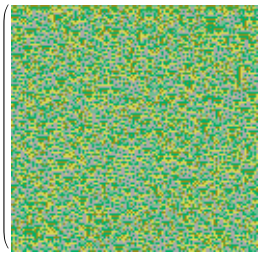
→ 10 033 010 686 584 540 → {4 → 0.156969, 3 → 0.843031},



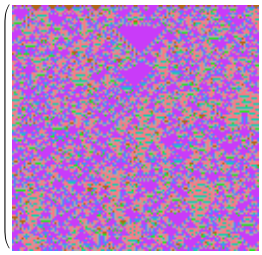
→ 7 330 900 949 575 011 → {4 → 0.132693, 3 → 0.866858},

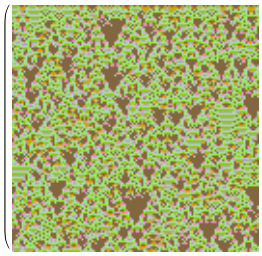


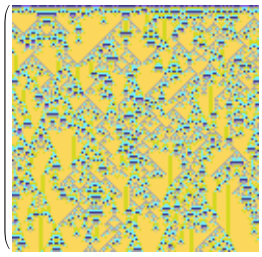
→ 2 943 550 672 701 792 → {3 → 0.116083, 4 → 0.883885},



→ 3 468 324 643 172 098 → {4 → 0.0991377, 3 → 0.900457},


→ 6 736 336 766 801 222 → {3 → 0.498866, 4 → 0.501134},

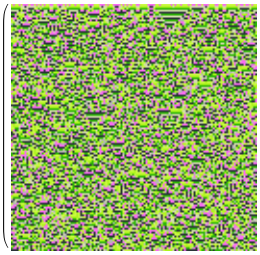

→ 1 466 636 402 463 332 → {3 → 0.218265, 4 → 0.781734},

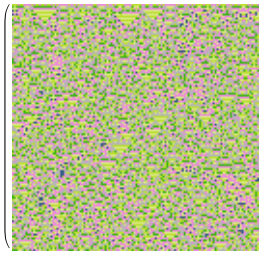

→ 1 428 111 228 778 929 → {3 → 0.0577248, 4 → 0.925458}}

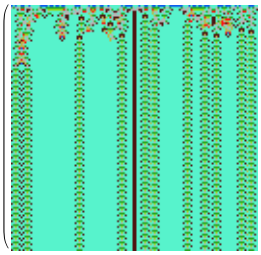
8-colour totalistic, range 1

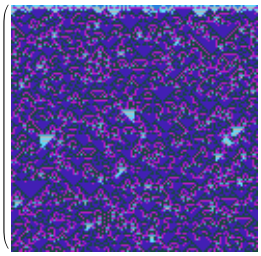
In[•]:=
```
test4Data8kr1C16 = data8TC[8, 128, 128];
Thread[
 test4Data8kr1C16 → netECA16[Keys@test4Data8kr1C16, {"TopProbabilities", 2}]]
```

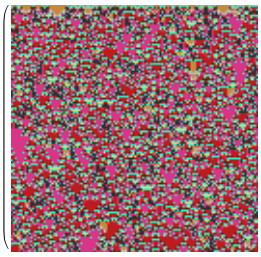
Out[•]=

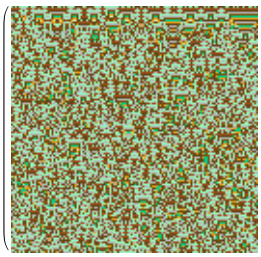

→ 42 020 590 261 131 320 575 → {4 → 0.0117574, 3 → 0.988243},

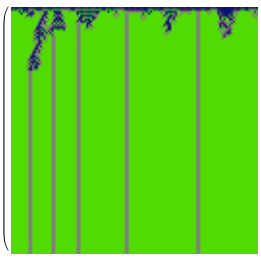

→ 12 610 707 415 579 864 791 → {4 → 0.0178909, 3 → 0.982109},

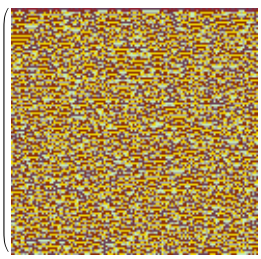

→ 15 365 143 935 027 257 744 → {1 → 0.0201417, 2 → 0.960127},


→ 20 096 665 423 769 445 584 → {4 → 0.0368129, 3 → 0.963187},


→ 15 686 501 956 369 591 456 → {3 → 0.111799, 4 → 0.88817},

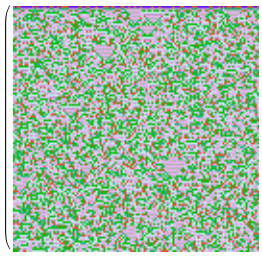

→ 46 724 218 297 933 137 114 → {4 → 0.123036, 3 → 0.876962},

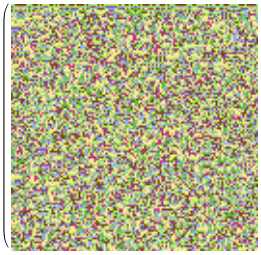

→ 44 175 329 969 224 582 408 → {4 → 0.0345722, 2 → 0.961729},

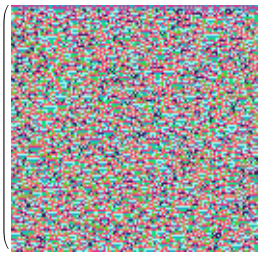

→ 65 643 433 669 628 134 604 → {4 → 0.355071, 3 → 0.643197}}

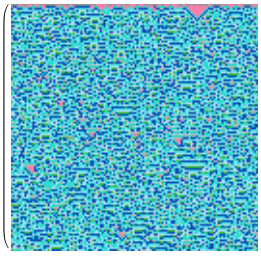
In[•]:=
```
test4Data8kr1C16 = data8TC[8, 128, 128];
Thread[
 test4Data8kr1C16 → netECA16[Keys@test4Data8kr1C16, {"TopProbabilities", 2}]]
```

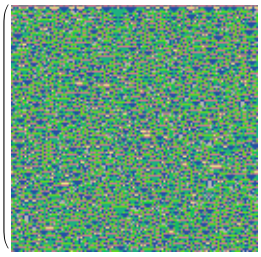
Out[•]=

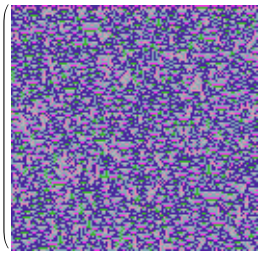

→ 36 507 401 174 866 450 102 → {3 → 0.244239, 4 → 0.75576},


→ 36 629 210 896 102 279 370 → {4 → 0.2431, 3 → 0.753856},

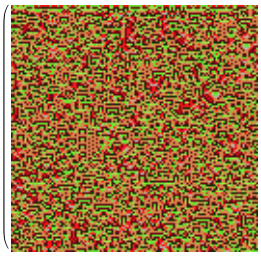

→ 50 599 399 972 817 837 073 → {4 → 0.24465, 3 → 0.75535},


→ 62 920 725 793 807 162 408 → {4 → 0.240564, 3 → 0.759435},

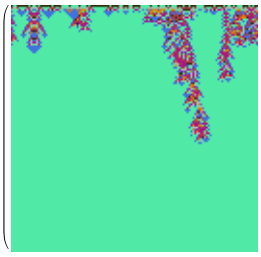

→ 37 059 387 199 693 125 873 → {4 → 0.0784766, 3 → 0.921523},



→ 16 229 780 842 690 144 670 → {4 → 0.0229385, 3 → 0.977062},



→ 41 966 635 277 478 833 399 → {4 → 0.243929, 3 → 0.755756},



→ 72 502 378 842 348 778 161 → {4 → 0.376498, 2 → 0.619948}}