Supplementary Materials: Deep Motifs and Motion Signatures

A. ARISTIDOU, D. COHEN-OR, J.K. HODGINS, Y. CHRYSANTHOU, A. SHAMIR

APPENDIX

We present the full specification of our convolutional neural network in Torch syntax. Note that, we use a modified version of FaceNet's nn4 network [Schroff et al. 2015] that is based on the GoogLeNet architecture [Szegedy et al. 2015] (the Inception model); its parameters are modified to work with our smaller size dataset (48×16).

```
local net = nn.Sequential()
   net:add(nn.SpatialConvolutionMM(3, 64, 7, 7, 1, 1, 3, 3))
   net:add(nn.SpatialBatchNormalization(64))
   net:add(nn.ReLU())
   net:add(nn.SpatialMaxPooling(3, 3, 1, 1, 1, 1))
   net:add(nn.SpatialCrossMapLRN(5, 0.0001, 0.75))

   net:add(nn.SpatialConvolutionMM(64, 64, 1, 1))
   net:add(nn.SpatialBatchNormalization(64))
   net:add(nn.ReLU())
   net:add(nn.SpatialConvolutionMM(64, 192, 3, 3, 1, 1, 1))
   net:add(nn.SpatialBatchNormalization(192))
   net:add(nn.ReLU())
   net:add(nn.SpatialCrossMapLRN(5, 0.0001, 0.75))
   net:add(nn.SpatialMaxPooling(3, 3, 1, 1, 1, 1))

   net:add(nn.Inception{
     inputSize = 192,
     kernelSize = {3, 5},
     kernelStride = {1, 1},
     outputSize = {128, 32},
     reduceSize = {96, 16, 32, 64},
     pool = nn.SpatialMaxPooling(3, 3, 1, 1, 1, 1),
     batchNorm = true
   })
   net:add(nn.Inception{
     inputSize = 256,
     kernelSize = {3, 5},
     kernelStride = {1, 1},
     outputSize = {128, 64},
     reduceSize = {96, 32, 64, 64},
     pool = nn.SpatialLPPooling(256, 2, 3, 3, 1, 1),
     batchNorm = true
   })
   net:add(nn.Inception{
     inputSize = 320,
     kernelSize = {3, 5},
     kernelStride = {2, 2},
     outputSize = {256, 64},
     reduceSize = {128, 32, nil, nil},
     pool = nn.SpatialMaxPooling(3, 3, 2, 2, 1, 1),
     batchNorm = true
   })
   net:add(nn.Inception{
     inputSize = 640,
     kernelSize = {3, 5},
     kernelStride = {1, 1},
     outputSize = {192, 64},
     reduceSize = {96, 32, 128, 256},
     pool = nn.SpatialLPPooling(640, 2, 3, 3, 1, 1),
     batchNorm = true
   })
   net:add(nn.Inception{
     inputSize = 640,
     kernelSize = {3, 5},
     kernelStride = {1, 1},
     outputSize = {256, 128},
     reduceSize = {160, 64, nil, nil},
     pool = nn.SpatialLPPooling(640, 2, 3, 3, 1, 1),
     batchNorm = true
   })
   net:add(nn.Inception{
     inputSize = 1024,
     kernelSize = {3},
     kernelStride = {1},
     outputSize = {384},
     reduceSize = {96, 96, 256},
     pool = nn.SpatialLPPooling(960, 2, 3, 3, 1, 1),
     batchNorm = true
   })
   net:add(nn.Inception{
     inputSize = 1024,
     kernelSize = {3},
     kernelStride = {1},
     outputSize = {384},
     reduceSize = {96, 96, 256},
     pool = nn.SpatialMaxPooling(3, 3, 1, 1, 1, 1),
     batchNorm = true
   })
   net:add(nn.SpatialAveragePooling(3, 3, 2, 2))
   net:add(nn.View(736))
   net:add(nn.Linear(736, opt.embSize))
   net:add(nn.Normalize(2))

   return net
```

REFERENCES

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*. IEEE Computer Society, Washington, DC, USA, 815–823.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*. IEEE Computer Society, Washington, DC, USA, 1–9.

2018. 0730-0301/2018/11-ARTXX $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn