Noname manuscript No.
(will be inserted by the editor)

Appendix of Learning a Perceptual Manifold with Deep Features for Animation Video Resequencing

Charles C.Morace · Thi-Ngoc-Hanh Le · Sheng-Yi Yao · Shang-Wei Zhang · Tong-Yee Lee *

Received: DD Month YEAR / Accepted: DD Month YEAR

Keywords deep learning · deep features · manifold learning · animation video resequencing

1 Training details of the autoencoder network for comparison

In our network architecture, we use both general convolutional layers and the concept of residual blocks [1]. A residual block is a block of layers where the input to the block is added element-wise to the output of the block. This technique helps prevent vanishing gradients which is a common problem in training deep neural networks. We use scaled exponential linear units (SELUs) [4] as our activation functions except in the last layer where we apply a sigmoid function to guarantee that the output image pixel values are between zero and one. We also use batch normalization layers [2] in our model to keep the values of tensors propagating in the network to have zero mean and unit variance. The benefits of the SELU activation functions and batch normalization layers are to train a deeper network and make training converge faster. Figure 1 gives additional details about the network architecture.

For training, we collect 20 Japanese cartoon animations, where each video is about 25 minutes long, and linearly scaled each frame to $w = 320$ pixels and a height $h = 180$ pixels to reduce training time. To avoid images which are nearly identical, we obtained the training images by uniformly sampling one out of every ten frames. In total, our training set and validation set consists of 60000 and 10000 images, respectively. We use L_2 distance to measures the

All authors are with Department of Computer Science and Information Engineering, National Cheng-Kung University, Taiwan
*Corresponding author and his email: tonylee@mail.ncku.edu.tw

error between the original images X and the reconstructed image $\phi(X)$:

$$L(X, \phi(X)) = \sum_{i=1}^{h} \sum_{j=1}^{w} \parallel X_{i,j} - \phi(X)_{i,j} \parallel^2, \tag{1}$$

where $X_{i,j}$ and $\phi(X)_{i,j} \in [0,1]^3$ are the red, green, and blue components of the pixel(i,j) in the original image and the reconstructed image, respectively. Then the optimal parameters of the encoder and decoder, are those which minimize a mean-square error loss across all iterations of the training process, where the batch size is set to 16. The initial parameters of the autoencoder, $\{\theta_0, \theta_0'\}$, are set by drawing samples from a truncated normal distribution similar to the technique described by Klambauer [4]. Finally, we use the stochastic gradient descent algorithm ADAM [3] to obtain the optimal solution.

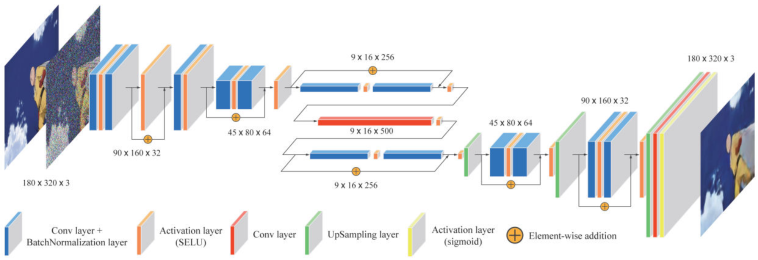

Fig. 1: The architecture of the denoising autoencoder used in testing animation reconstruction.

2 Traditional manifold learning algorithms

2.1 Isomap

The Isomap algorithm has three steps. The first step is to determine the neighbors of each image and represent these relations as a weighted graph G. In our evaluation, we use the k nearest neighbors with edge weights equal to the L_2 distance of neighboring images in image space. The second step is to estimate the geodesic distance $d_M(i,j)$ between all pairs of points by computing their shortest path distance $d_G(i,j)$ in the graph G. The final step applies classical multidimensional scaling [5] to the matrix of geodesic distances $D_G = \{d_G(i,j)\}$ to obtain the set of coordinates vectors Y.

2.2 Locally Linear Embedding

The locally linear embedding (LLE) algorithm recovers global structure of a nonlinear manifold from locally linear fits. Like Isomap, the first step of the

LLE algorithm is to determine the neighbors of each image. Again, we use the k nearest neighbors measures by L_2 distance in image space. Again, we use the k nearest neighbors measured by L_2 distance in image space. The second step is to determine an optimal set of weights $w_{i,j}$ such that the reconstruction loss $\varepsilon(W) = \sum_i (x_i - \sum_j w_{i,j} x_j)^2$ is minimized subject to two constraints. First, $w_{i,j} = 0$ if x_i and x_j are not neighbors, and second, $\sum_j w_{i,j} = 1$. After the optimal weights have been found, the set of coordinates Y are obtained by minimizing the function $\phi(Y) = \sum_i (y_i - \sum_j w_{i,j} y_j)^2$.

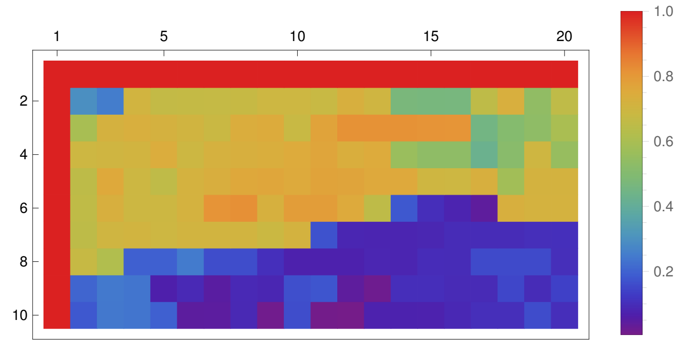

Fig. 2: Comparison of LLE errors for different embedding dimension and number of nearest neighbors.

3 LPIPS evaluation

We present here some additional details and results of the animation reconstruction experiment. Figure 2 shows the error rates for a single test case using the LLE metric and different parameter settings for the embedding dimension and number of nearest neighbors. Note that very few parameter settings result in low error rates. This highlights the fact that traditional manifold learning technique requires a lot of parameter tuning to produce good results.

4 Results

To consider an input animation as ground truth for a Hamiltonian path sequence, it must not contain cyclic motion. Thus we visually inspect each test animation and remove test cases with obvious cyclic motion. Additionally, we remove trivial cases where all test methods perfectly reconstruct the animation. In total, we tested the reconstruction of 39 animations.

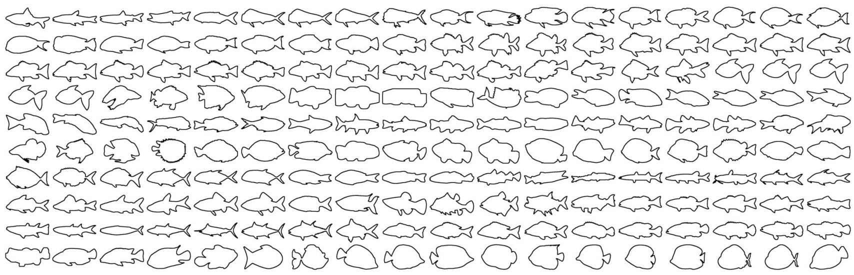

Fig. 3: Grid image layout example generated by the proposed method.

About the additional application of our framework, we show in Figure 3 an example of a grid image layout generated by the proposed Hamiltonian path sequencing method applied to a collection of fish contour images.

References

1. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
2. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
3. D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
4. G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30:971–980, 2017.
5. J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.