A. AUTOENCODER RECONSTRUCTION DETAILS

Convolutional autoencoders were trained on 96 $\times$ 96 images. The convolutional network architecture consists of 3 convolutional layers, each with a filter size of 5 and a stride of 2. This means the input is reduced from (96,96) to (48,48), then to (24,24), and finally to (12,12). The first layer has 128 filters, the second 256, and the final 512. All layers have ReLU activations, except the bottleneck and output, which are tanh. The deconvolutional layers mirror the convolutional layers and are implemented as convolutional layers that are preceded by an upsampling step that creates a layer with 2 times the dimensions of the input layer by repeating the values of the input. To explore the role of the capacity of the convolutional layer, we built models with bottlenecks of both 128 and 512 units. However, we report only results for the 128-unit network as qualitative performance of the 512-unit network was quite similar.

We train our convolutional autoencoders on the 100,000 images in STL-10 referred to as the "unlabeled" set, and of the remaining data, we formed a hold-out set of 10,400 images, which was used to determine when to stop training. Other than early stopping, no regularization was used during training. We used the "Adam" optimizer [31].

B. CLASSIFICATION DETAILS

For the classification experiments, we resized the Extended Yale B [23] images to 48×48 and split the data randomly into a training, validation, and test set in a 60%-20%-20% ratio. We trained convolutional autoencoders with the same architecture described in Section A except here we used a 32-unit bottleneck with ReLU activations and batch normalization [32] on all layers except the output layer of the decoder. Batch sizes of 32 and Adam [31] with a learning rate of 0.001 were used to train the networks. After training each of the autoencoders to convergence on the training set, we extracted bottleneck representations for the training and validation sets. We trained a SVM with a linear kernel to predict identity and SVR with RBF kernels to predict azimuth and elevation. Hyperparameters of the SVMs were selected via three-fold cross-validation on the training plus validation set. For azimuth and elevation prediction, we use SVR with RBF kernels and use grid search to select $C \in \{1, 10, 100, 1000\}$. For identity classification, we use SVM with a linear kernel and select $C \in \{0.01, 0.1, 1, 10, 100, 1000, 10000\}$. The grid search was performed via three-fold cross-validation on the training plus validation set.

C. IMAGE SUPER-RESOLUTION DETAILS

For the super-resolution experiments, all input images are converted from RGB to YCbCr color space and only Y channel is used for training and testing. To generate the training and testing data, we use bicubic method to perform downsampling. For visualizing color images, we first upsample the Cb and Cr channels using bicubic, merge the result of Y channel and then convert it back to RGB color space. The above procedure is the same as SRCNN [24] which is the common practice in single image super-resolution literature. Moreover, we use stochastic gradient descent (SGD) with momentum for optimization. The learning rate is fixed to 0.001 and momentum is 0.9. We did not use any weight decay since the model is fairly simple and we did not observe any overfitting. For training the MS-SSIM loss, we use 5 scales and downsample the image with ratio 2 for each one. Fig. 3 provides close-up visual illustrations.

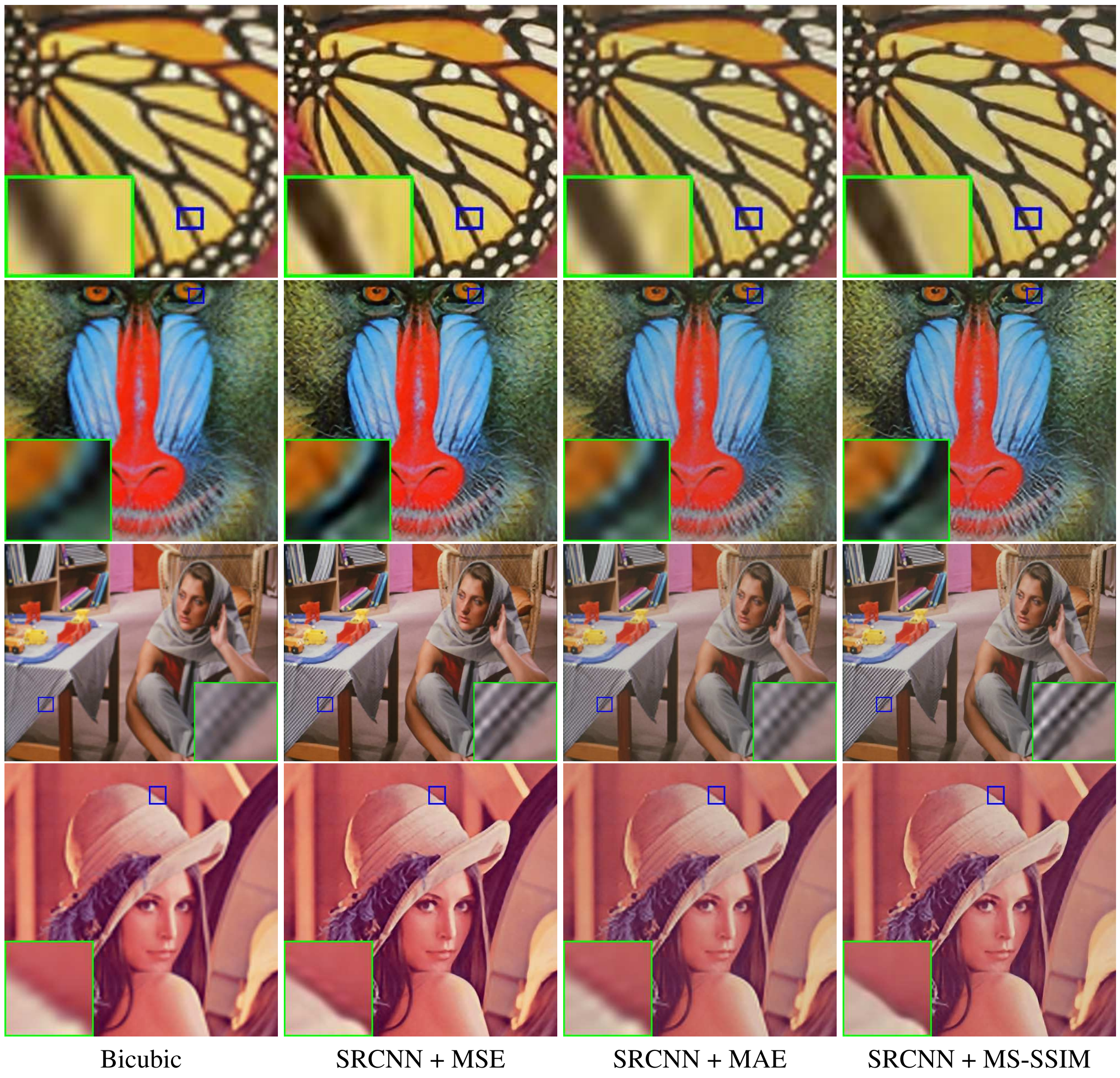

Fig. 3. Visual comparisons on super-resolution at a magnification factor of 4. MS-SSIM not only improves resolution but also removes artifacts, *e.g.*, the ringing effect in the bottom row, and enhances contrast, *e.g.*, the fabric in the third row.