CS230

Face Recognition with Sub-Sampled Images

Chung Chun Wan
cwan@stanford.edu

Yuxuan Wang
ywthree@stanford.edu

Abstract

Deep learning (DL) has proven to be a power paradigm in the field of imaging and computer vision. Many DL-based techniques have been developed to upsample images for human consumption. In this work, we have developed a CNN-based model that optimizes instead for another DL mode, the FaceNet. Our model takes as input aggressively sub-sampled images (⅛ of the original resolution in both imaging dimensions). We demonstrate that our model outperforms several other non-DL based upsampling filters in certain dataset. We identify the areas of improvement of our model and insights obtained from this work. If this technology is developed successfully, it can extend the application range of the FaceNet model for a given camera hardware configuration..

1 Introduction

Given a particular sensor resolution and face recognition block, there is a tradeoff between the camera's field of view and the working distance as illustrated in Figure 1 below depending on the input image size requirement of the face recognition block.

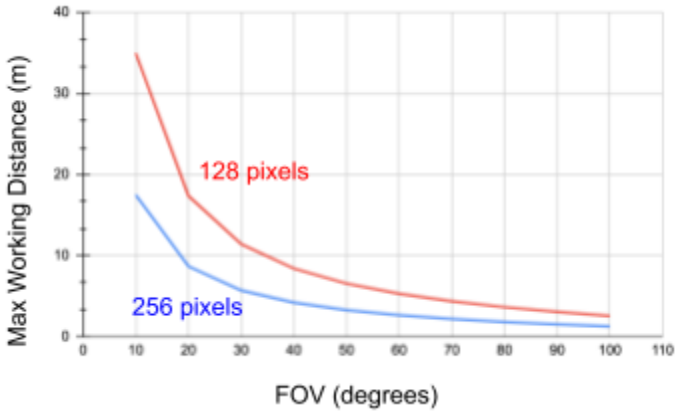


Figure 1: Working distance vs Camera Field-Of-View (FOV) assuming human face length is 12cm and sensor resolution along the direction of interest is 3264 pixels (typical horizontal resolution for a 8M-pixel sensor).

It is always desirable to have a wider FOV while, at the same time, a longer working distance (red curve). One way to achieve this goal is to use a sensor with higher resolution but it will increase cost and size of the camera. In this project, we explore a DL-based approach such that given a particular face recognition block, e.g. FaceNet, we can significantly reduce the input image size (e.g. from NxN to N/8xN/8) while matching as much as possible the original accuracy.

2 Related work

The neural network based image upsampling or super-resolution aims to recover high-resolution (HR) images from the low-resolution (LR) images[1, 2]. Typically, the loss functions used in these learnings measure the visual

differences between the recovered HR image and the ground-truth HR image, which could be roughly categorized into 2 classes, the pixel loss[3, 4] and texture loss[5, 6]. A lot of different architectures and training strategies have been investigated and . For example, ResNet and DenseNet based networks[7, 8, 9] have made very great achievements, and Generative Adversarial Nets (GAN) approaches[10, 15] show very promising results as well. However, these networks were mostly designed to optimize the upsampled images from the perspective of human beings.

For machine vision applications, the conventional image quality metrics may not be as important as to the human vision. For example, for the face recognition application, the FaceNet is mainly concerned about the information in the face region and ignores the information from other areas. In this paper, motivated by the machine vision applications, we try to build a Unet-based convolutional neural network for image super-resolution to serve machine vision the best, in our case, the FaceNet.

3 Dataset and Features

Figure 2 shows the workflow of our project. Major decisions we have to make include what FaceNet model we should use and how we could generate the (x,y) pairs for the training, dev, and test sets.

We have decided to use the open-source implementation of Facenet in Keras by Hiroki Taniai[11]. This model requires a (160,160,3) input image size to generate a 128-d embedding vector. This FaceNet is implemented using the Inception ResNet v1 and is considered to be superior than the very original FaceNet[12].

Next, we have decided to use the Labelled Face in the Wild (LFW)[13] as our raw image dataset. The dataset contains 13,233 images of faces collected from the web. Each face has been labeled with the name of the person pictured. Each image has a size of (250,250,3). Figure 3 shows some example pictures of this dataset.

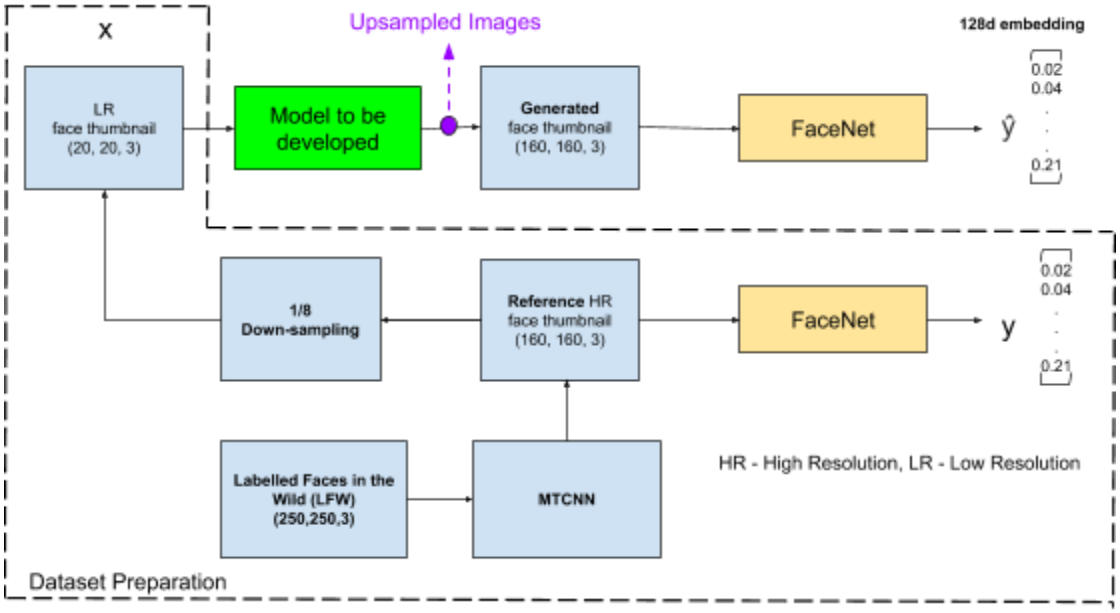


Figure 2: Project workflow

Figure 3: Example pictures of the LFW dataset (Left- Zhang Ziyi, Middle -Yao Ming, Right - Bill Gates)

The training set, dev set, and test set consist of 11910 pictures (~90%), 972 pictures (~7.35%), and 351 (~2.65%), respectively. This raw dataset is fed into the Multi-task Cascaded Convolutional Neural Networks (MTCNN)[14] to create images of cropped faces with size (160,160,3). These cropped images are fed to the FaceNet to generate the embeddings which are our ground truth (y). The same images are downsampled by a factor of 8 using the image resizing function provided by the Python Imaging Library (PIL) to create the input X that has a size (20,20,3). Figure 4 shows an example of a high-resolution (HR) and the corresponding low-resolution (LR) image pair.

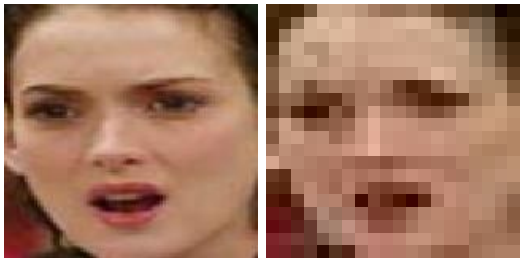


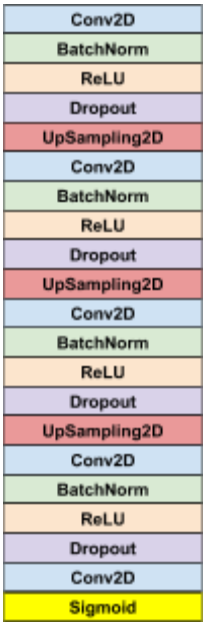
Figure 4: High-resolution (HR) and low-resolution (LR) image pair (⅛ the resolution in both image dimensions)

4 Methods

Loss Function: The goal of this project is to develop a model that converts low-resolution images, with sizes (20,20,3), to high-resolution ones that have sizes (160,160,3). Instead of forcing the model to generate images that appeal to humans, we let the model learn what the most optimal things should be in order to match the embeddings y and ŷ. A natural choice of the loss function is the mean square error.

$$Cost\ Function\ J = \frac{1}{n}\sum_{i=1}^{n} ||y - \hat{y}||^2$$

Architecture: Figure 5 shows our final model architecture. ReLU is used as the activation functions except the last layer where a sigmoid is used. The reason to use a sigmoid at the last layer is to confine the output values to be between 0 and 1 as the output of this model is supposed to be an image. Batch normalization layers are used to enhance training and dropout layers are used for regularization. To perform upsampling, one architectural choice we have made is to use Upsampling2D() followed by a Conv2D() layer instead of a single Conv2DTranspose() layer because it has been shown in the literature that the later approach would result in artifacts in the output images.



Layer (type)	Output Shape	Param #
model_input_UR (InputLayer)	(None, 20, 20, 3)	0
decoder_conv_t_0_UR (Conv2D)	(None, 20, 20, 256)	7168
batch_normalization_1 (Batch	(None, 20, 20, 256)	1024
activation_1 (Activation)	(None, 20, 20, 256)	0
dropout_1 (Dropout)	(None, 20, 20, 256)	0
up_sampling2d_1 (UpSampling2	(None, 40, 40, 256)	0
decoder_conv_t_1_UR (Conv2D)	(None, 40, 40, 128)	295040
batch_normalization_2 (Batch	(None, 40, 40, 128)	512
activation_2 (Activation)	(None, 40, 40, 128)	0
dropout_2 (Dropout)	(None, 40, 40, 128)	0
up_sampling2d_2 (UpSampling2	(None, 80, 80, 128)	0
decoder_conv_t_2_UR (Conv2D)	(None, 80, 80, 64)	73792
batch_normalization_3 (Batch	(None, 80, 80, 64)	256
activation_3 (Activation)	(None, 80, 80, 64)	0
dropout_3 (Dropout)	(None, 80, 80, 64)	0
up_sampling2d_3 (UpSampling2	(None, 160, 160, 64)	0
decoder_conv_t_3_UR (Conv2D)	(None, 160, 160, 32)	18464
batch_normalization_4 (Batch	(None, 160, 160, 32)	128
activation_4 (Activation)	(None, 160, 160, 32)	0
dropout_4 (Dropout)	(None, 160, 160, 32)	0
decoder_conv_t_4_UR (Conv2D)	(None, 160, 160, 3)	867
activation_5 (Activation)	(None, 160, 160, 3)	0

Total params: 397,251
Trainable params: 396,291
Non-trainable params: 960

Figure 5: Final Model Architecture

Optimizer and Transfer Learning: Adam is used as the optimizer in this project. Also, mini-batch with a batch size of 128 is used. Transfer learning is used in the sense that we freeze all the layers within the FaceNet model, so during training, we are only optimizing the parameters of our model.

Other Hyperparameters: The two key hyperparameters we have explored are number of epochs and number of filters of the intermediate layers. We have selected the final number of epochs when we reached the lowest loss in the dev set, a technique called early-stopping. In terms of number of filters, we have experimented using a simpler and more complicated model. We again made our final decisions based on the performance of the model on the dev set.

5 Results

Presentation of results: After we have optimized our model using the cost function on the training, dev, and test sets, we are mostly interested in evaluating the performance of our model against three other non-DL approaches, including the Image.NEAREST, Image.BILINEAR, and Image.BICUBIC filters available in PIL.

The metric we used in this comparison is the L2 distance between the embeddings of the upsampled images to the ground truth embeddings of the original high-resolution images. A model is considered better if the L2 distance from the ground truth is shorter. Figure 6 shows a summary of our results. In each of the three tables, the top row indicates the number of samples our model performs better against the three other up-samping filters while the bottom row indicates otherwise.

	NEAREST	BILINEAR	BICUBIC
Better	10991	6999	6691
Worse	919	4911	5219

Training Set (Total Images = 11910)

Better	861	496	467
Worse	111	476	505

Dev Set (Total Images = 972)

Better	302	166	167
Worse	49	185	184

Test Set (Total Images = 351)

Figure 6: Summary of comparisons between our model and alternative upsampling filters
(Top - Training Set, Middle - Dev Set, Bottom - Test Set)

Our model performs much better than the NEAREST filter regardless of which dataset images are from. It performs better than the BILINEAR filter for data from both the training and dev set but slightly worse than those from the test set. Lastly, our model performs better than BICUBIC filter for images from the training set but generally worse for those from the dev and test sets.

Analysis of results: The results are not surprising to us, especially the part that our model performs better than the other three filters in the training set. It is also intuitive that it is easier to beat the performance of NEAREST regardless of the data set. BILINEAR is performing a linear interpolation in a 2x2 environment while that of the BICUBIC is a cubic spline interpolation in a 4x4 environment. Given that our model only uses filters with 3x3 kernel size, that may explain why our model is in general performing poorer than BICUBIC.

As discussed earlier, our approach is not to force the model to learn to produce visually pleasing images. Our only requirement is to generate images with embeddings that are closer to the ground truths. Figure 7 shows two

examples, one that our model performs better while the other the BICUBIC filter performs better. Not completely conclusive but it seems our model has learned to generated visually pleasing images regardless indicating that FaceNet has learned to generate embeddings based on image qualities.

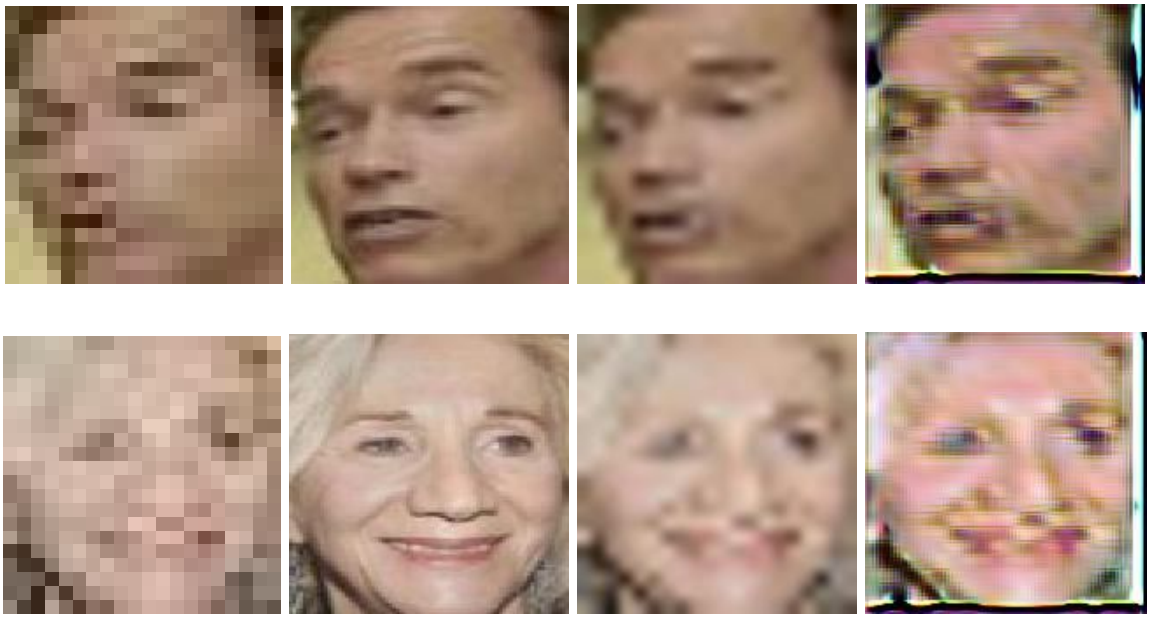


Figure 7: Positive and negative examples. Top - our model performs better. Bottom - BICUBIC performs better. From left to right, input image, ground truth HR image, BICUBIC upsampled image, upsampled image using our model

6 Conclusion/Future Work

In this project, we have developed a DL model that up-samples aggressively sub-sampled images (⅛ the resolutions in both image dimensions) before they are fed to the FaceNet. Our CNN-based model outperforms some common upsampling filters, such as NEAREST, BILINEAR, and BICUBIC, in the training set. Compared with the BICUBIC filter, our model does not perform as well in the dev and test sets. We hypothesize that the main reason to be be that our model uses only filters with 3x3 kernels.

Future work includes experimenting with different model architectures, ones that use larger (e.g. 5x5) filters, U-net, or inception network. An interesting direction would be to formulate the problem in the GAN framework treating our model as the generator while the FaceNet as the discriminator. Once an improved model has been developed, one can also analyze the upsampled images which would provide additional insights on what exactly the FaceNet is looking for.

7 Contributions

Chung Chun and Yuxuan worked together the entire time on developing, training, and testing the model. Chung Chun and Yuxuan focused on the write-up of this final report and poster, respectively.

Source code is available @ https://github.com/wancc3/CS230-Final-Submission

References:

[1] Z. Wang, J. Chen, and S. C. Hoi. Deep learning for image super-resolution: A survey. arXiv:1902.06068, (2019).
[2] Yang, W., Zhang, X., Tian, Y., Wang, W., Xue, J.H.: Deep learning for single image super-resolution: A brief review. arXiv preprint arXiv:1808.03344 (2018).
[3] N. Ahn, B. Kang, and K.-A. Sohn, “Fast, accurate, and lightweight super-resolution with cascading residual network,” in ECCV, 2018.
[4] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, “Enhanced deep residual networks for single image super-resolution,” in CVPRW, 2017.
[5] M. S. Sajjadi, B. Scholkopf, and M. Hirsch, “Enhancenet: Single ¨ image super-resolution through automated texture synthesis,” in ICCV, 2017.
[6] L. Gatys, A. S. Ecker, and M. Bethge, “Texture synthesis using convolutional neural networks,” in NIPS, 2015.
[7] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, “Low-complexity single-image super-resolution based on nonnegative neighbor embedding,” in BMVC, 2012.
[8] R. Timofte, V. De Smet, and L. Van Gool, “A+: Adjusted anchored neighborhood regression for fast super-resolution,” in ACCV, 2014.
[9] T. Tong, G. Li, X. Liu, and Q. Gao, “Image super-resolution using dense skip connections,” in ICCV, 2017.
[10] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, ´ A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang et al., “Photorealistic single image super-resolution using a generative adversarial network,” in CVPR, 2017.
[11] https://github.com/nyoki-mtl/keras-facenet
[12] F. Schroff, D. Kalenichenko, and J. Philbin. “Facenet: A unified embedding for face recognition and clustering”. In Proc. CVPR, 2015.
[13] http://vis-www.cs.umass.edu/lfw/
[14] Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multi-task cascaded convolutional networks. arXiv preprint arXiv:1604.02878 (2016).
[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. “Image-To-Image Translation With Conditional Adversarial Networks”. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1125-1134