Feature Blending with Variational Autoencoders and Generative Adversarial Networks

Carlo Salomon Ruiz, Lahav Lipson
Columbia University
{csr2131, lol2107}@columbia.edu
github.com/Lahav174/face-merger

Abstract

We propose a new method for image blending that combines two images on the latent feature space rather than on the pixel space. To achieve this, we rely on a convolutional autoencoder (AE) to produce both an encoder that can generate a compressed feature representation and a decoder that can recover an image from the compressed feature space. Blending occurs at the feature representation level (i.e. the output of the encoder). We then use the decoder to recover the image resulting from the blended encodings. We introduce a Generative Adversarial Network (GAN) to refine the encoder and decoder so the result is more realistic.

1. Introduction

Image blending is a standard problem in Computer Vision with many applications. Historically, blending has been done in pixel space, that is, the output image is a linear combination of the input images' pixel values. Images can be blended at different resolutions, or even blended via combining lower frequency components and then adding higher frequencies (blending in Fourier space). Here we propose a considerably different blending technique; blending based on the latent feature space. Our goal with feature blending is to produce more realistic images.

Image blending is a complex task due to the fact that blending images in a realistic manner often depends on the content of the images. For instance, a blended image of two elliptically shaped fruits would be one where the resulting shape is still an ellipse. One method which has historically been used to accomplish realistic blending is the Laplacian-pyramid [9], which combines the images at lower frequencies, and then applies the higher frequency components. While Laplacian-pyramids can be used to create realistic image blendings, such methods do not take the semantics of the content of the image into account. For instance, if we are blending two images of faces, we would want the output to be a face which a human without any irregularities could posses (e.g. ears are symmetric, hair is a single color, eyes are the same color and symmetric, etc). To resolve this, we may want to apply an auxiliary constraint to the blending.

In order to develop an image-blending system which can intelligently account for these features in the image, we propose to train a neural network on a specific dataset with the goal of being able transform images to-and-from a latent space where images of that dataset can be combined (linearly) in a meaningful fashion. We intend to accomplish this by training an autoencoder to preserve the input as best as possible, with the additional objective that the encoder and decoder components be able to convert to and from the aforementioned latent space, respectively.

1.1. Related Work

Autoencoders have a long history in machine learning literature. They are often used to generate a latent vector representation of an input image that selectively preserves the most important features, similar to other dimensionality reduction techniques like Principal Component Analysis (PCA). Consequently, high frequency components of the input are often discarded, resulting in blurry output images. Variational Autoencoders (VAEs) were introduced in [1] to mitigate the issue of blurry output common in the standard autoencoder architecture. Rather than passing the output of encoder directly to the decoder, VAEs build a probability distribution from the encoder output and sample from that distribution to generate input for the decoder. This method has shown to decrease the blurriness of the output of the autoencoder.

More recent work has combined GANs with AEs to further constrain the latent feature space [4]. In this work, Makhazani et al. use a GAN to encourage the aggregate posterior of the latent space to follow an arbitrary distribution, in their case a Gaussian distribution, in order to promote aesthetic behavior when drawing randomly from the distribution. In this paper, however, we explore the effect of

encouraging a specific posterior of the latent space by training an adversarial network on the decoded latent vectors, as opposed to their latent form.

In addition to the blur induced by the small latent feature space, one issue with generative networks is that often the output will be blurry due to the use of per-pixel loss functions like L_2 or binary cross entropy that strongly penalize sharp images that are translated slightly relative to their label. To resolve this, researchers have employed a technique called perceptual loss, where auxiliary neural networks are used in order to extract features from the image output, and these learned features are used to compute the loss [6]. Similarly, Goodfellow et al. [3] showed the ability of an adversarial discriminator network to provide gradients on which to train the generator, resulting in the generator being able to produce lifelike imagery.

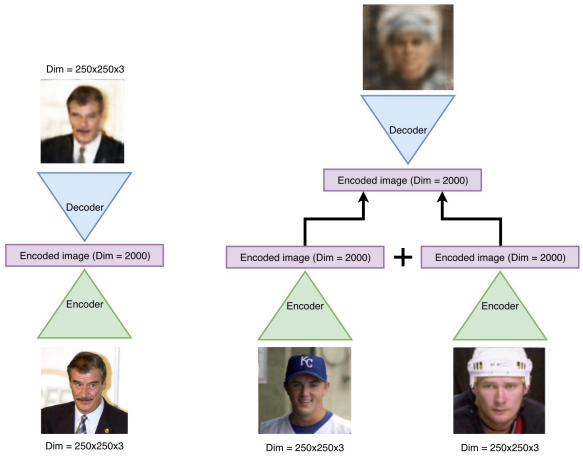

Figure 1. Results we achieve with a VAE alone

2. Method

We attempt to train a convolutional variational autoencoder with two objectives:

1. For the trained AE to be able to reconstruct its input as accurately as possible

2. To constrain the latent feature space such that one can linearly combine two latent vectors in order to get a 'merged' image that looks like both

We first attempt to satisfy objective one, acknowledging that blurry output is one of main challenges of working with autoencoders and an obstacle for generating recognizable, aesthetic images. In order to compensate for this limitation, we employ an adversarial network with the objective of discriminating between the output of the autoencoder and the training data. Our expectation is that by using a weighted combination of construction loss and adversarial loss, the gradients will encourage the decoder to artificially add high frequencies to the output image in order to fool the discriminator. Figure 1 shows the baseline which we wish to improve.

Once the autoencoder is able to sufficiently reproduce the output, we augment the model architecture to also satisfy the second objective. We use an alternating scheme: For every two batches a and b, we process a as we would any batch when training the autoencoder to reproduce its input. Then, in order to train the network to additionally satisfy the second objective, we pass both batches a and b through the encoder, average the two vectors, and then pass the result through the decoder to get the final output. We then use the discriminator to encourage this 'merged' output to have similar features to the training data, a similar approach as Goodfellow et al. [3]. When satisfying the 1st objective, we use a mostly reconstruction loss, with a proportionally small amount of adversarial loss. Since our goal is for the output of our merging to be not be too similar to either input but instead simply be realistic (i.e. similar to the training data), we use 100% adversarial loss when training for the second objective.

2.1. Dataset

For our dataset, we used the Toronto Face Dataset (TFD). The dataset consists over just over 13,200 images, each with dimensions $250 \times 250 \times 3$. This resolution is high enough that the facial features are clearly identifiable to the human eye, making it easy for us as developers to qualitatively evaluate the success of our model, but also not high enough definition that we cannot use large batch sizes efficiently. One pitfall of this dataset is that 13,200 images is not as large as other datasets, which may force the network to have to extrapolate more often.

2.2. Generator

For our generator, we began testing our results with a standard autoencoder. The results were somewhat blurry, as expected. Increasing the size of the latent variable space yielded slightly less blurry results, although not much change.

We then added convolutional layers and increased number of filters in an effort to mirror VGG architecture [10]. This got us slightly better results, which motivated us to actually implement VGG11 as the encoder, initializing the weights to some pre-trained values. Despite our high expectations for this method, our results actually worsened and training time increased substantially. We believe the improvement in the encoder was mitigated by the deeper decoder, which had to be trained from scratch and given that it's a deep network, it required a much larger dataset than TFD.

We then configured our network architecture as a variational autoencoder, and noticed slightly less blurry

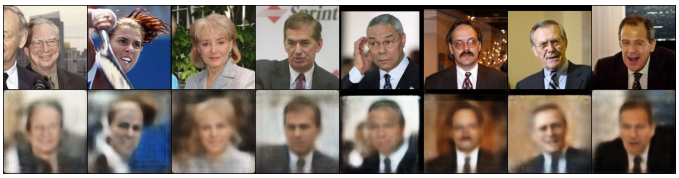

Figure 2. Variational Autoencoder results

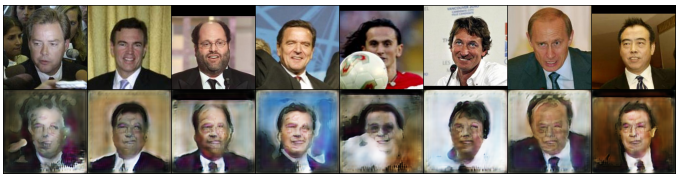

Figure 3. Sharp images resulting from partial use of discriminator loss

results, although the resulting images were still far from ideal. See Figure 2. After many different architectures, we had the best results with the following architecture for the encoder of the autoencoder:

Encoder network architecture

Convol.	Output channels: 8	Kernel width: 7
Convol.	Output channels: 8	Kernel width: 7
Batch norm.		
ReLU		
Pooling	Width: 2	
Convol.	Output channels: 16	Kernel width: 9
Convol.	Output channels: 16	Kernel width: 9
Batch norm.		
ReLU		
Pooling	Width: 2	
Convol.	Output channels: 32	Kernel width: 9
Convol.	Output channels: 32	Kernel width: 9
Batch norm.		
ReLU		
Pooling	Width: 2	
Full. Conn.	Width: 10,368	Output size: 2000

As one can see, the architecture is separated into three similar chunks. The result of a $250 \times 250 \times 3$ facial image passed through this network is our latent encoding. To decode this latent vector, the variational autoencoder will use this vector as input for two different feed-forward layers that preserve length, corresponding to the mean and std-dev of the latent probability distribution, respectively. Then, a new vector of length 200 is created by sampling values from the aforementioned distribution, and then this vector is passed through another convolutional network very similar to the encoder. The convolutional network for the decoder is the mirror image of the encoder architecture, with the pooling and activation coming after the deconvolution in each chunk as opposed to before in the encoder. For the output of the network, we use a sigmoid activation function.

2.3. Discriminator

The discriminator network has the same architecture as that of the encoder described above, with the exception of three fully connected layers and three activation functions appended directly after the final pooling layer:

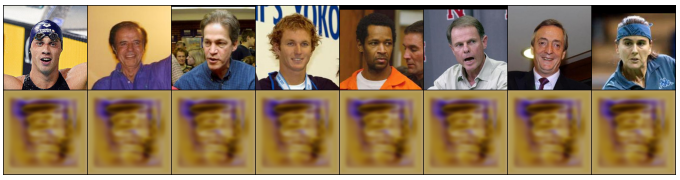

Figure 4. Total mode collapse in the output

Full. Conn.	Width: 10,368	Output size: 250
ReLU		
Full. Conn.	Width: 250	Output size: 80
ReLU		
Full. Conn.	Width: 80	Output size: 1
Sigmoid		

2.4. Loss

Our setup included two types of loss:

- Reconstruction loss. This is used when the AE must reproduce its input. For this, we used a simple MSE loss, as is standard for image reconstruction tasks.

- Adversarial loss. This loss, provided by the discriminator network, is used when we are training the AE to produce quality images from merged latent vectors. This loss is also used in combination with the reconstruction loss when the AE is trained to reproduce its input. We experimented with different weightings of losses, such as $0.1 \times advLoss + 0.9 \times reconLoss$. Originally, our adversarial loss was calculate by comparing the output of the discriminator network to a vector of 1s using binary cross entropy. To further improve the model, we modified our discriminator to use the Wasserstein distance, aka a Wasserstein GAN [5]. Using this loss metric makes our discriminator continuous and differentiable everywhere, allowing us to train the discriminator very strongly without having to worry about vanishing gradients. By training the discriminator strongly, we can provide better gradients for training the generator. The only modification we needed to make to the discriminator network architecture is the removal of the final sigmoid activation.

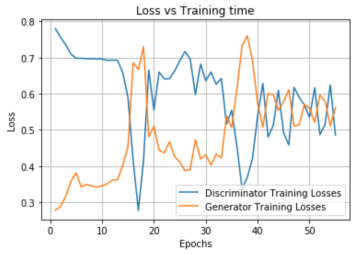

Figure 5. Oscillations in Discriminator vs Generator Loss

3. Results

Our objective in this project was to first be able to have the autoencoder produce sharp, realistic reconstructions of its input. As expected, our original VAE produced blurry images. Increasing the width of the latent vector increased quality slightly, although the output could only get results. See Figure 2. After applying the GAN to sharpen the output images, we observed artificial edges being applied to the output. Unfortunately, the network couldn't always intelligently figure out where to insert the edges, so we observed many disfigured faces in our results. See figure 3.

A large part of our experimentation involved playing around with different learning rates and weightings of reconstruction loss and adversarial loss. Using 1% to 10% adversarial loss (with the remaining being reconstruction loss) resulted in sharp but somewhat inaccurate results as seen in Figure 3. When we further increased the adversarial loss, we often noticed partial mode collapse, with many images belonging to one of a handful of different apparent categories. With more than 20% adversarial loss, we noticed stronger mode collapse, with the output images being identical across all inputs. This ended up being the dominant issue during our project. See Figure 4. By looking at our loss graphs, we predicted that this frequent mode collapse was due to the discriminator learning too quickly. By keeping the learning rates at values close to $1e10^{-5}$, we observed other phenomena such as oscillations between the discriminator loss and generator loss. See Figure 5.

One interesting consequence of the oscillating losses was that the generator almost began to succeed in generating facial images before the mode collapsed after about 50 epochs. Before epoch 30, the generator could only produce noise. Afterwards, it could resolve sharp facial images (although still far from perfect), and then after epoch 50 the mode collapsed. See figure 6.

Ultimately, our training scheme wasn't working. To prevent mode collapse, we implemented a Wasserstein GAN (WGAN) as previously mentioned. This approach did not have the same mode-collapse issues as before, although our setup was still unable to resolve sharp images even when we used a significant amount of adversarial loss. Another side

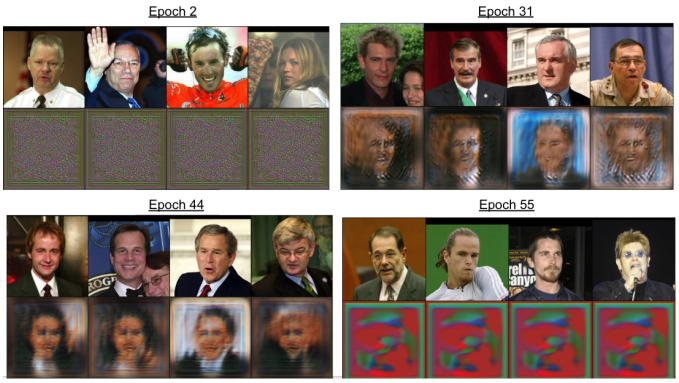

Figure 6. Unstable behavior created by oscillating losses

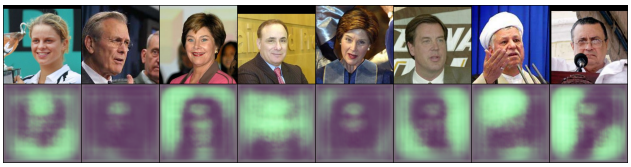

Figure 7. Output produced using Wasserstein loss

Figure 8. Loss graphs from the WGANs.

effect was that the output images were always green. See Figure 7 & 8. Issues relating to our WGAN, especially the green effect, are likely not so difficult to resolve, though we ran out of time and credits to run further experiments.

4. Future Improvements and Other Findings

To improve our model, we would continue experimenting with our Wasserstein GAN to see if we can mitigate the issues. We may also employ various additional methods such as multi-scale convolution, where we concatenate the output of each convolution layer with the output of the previous layer. We may could use Feature Matching and minimize the statistical difference between the features of the input images and those of the generated images. We could also use minibatch discrimination to penalize the generator for constant output. Lastly, we may employ one-sided label smoothing to prevent overconfidence in the generator. We were unable to produce satisfactory reconstructions with our proposed architecture so we did not get to actually average two embeddings and see the decoded results.

References

[1] Kingma et. al, *Auto-Encoding Variational Bayes*, 2013

[2] Masci, Jonathan, et al. *Stacked Convolutional Auto-encoders for Hierarchical Feature Extraction*, International Conference on Artificial Neural Networks. Springer, Berlin, Heidelberg, 2011.

[3] Goodfellow, Ian, et al. *Generative Adversarial Nets* Advances in neural information processing systems. 2014.

[4] Makhzani, Alireza, et al. *Adversarial Autoencoders*, 2016

[5] Wu, Huikai et al, *GP-GAN: Towards Realistic High-Resolution Image Blending*, 2017

[6] Johnson J., Alahi A., Fei-Fei L. (2016) Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9906. Springer, Cham

[7] Arjovsky, Martin, Soumith Chintala, and Lon Bottou. "Wasserstein gan." arXiv preprint arXiv:1701.07875 (2017).

[8] Huang, Gary B., et al. "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments." Workshop on faces in'Real-Life'Images: detection, alignment, and recognition. 2008.

[9] Burt, Peter J., and Edward H. Adelson. "The Laplacian pyramid as a compact image code." Readings in Computer Vision. 1987. 671-679.

[10] Simonyan, Karen et. al. *VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION*, 2014