Lab 9 : Normalizing flows on the Olivetti Faces dataset

Pau Hidalgo Pujol - APRNS @ GIA-UPC

December 6, 2024

1 Introduction

The Olivetti Faces dataset [1] consists of 400 grayscale images of size 64×64, representing 40 individuals with 10 images per person. Due to the high dimensionality of the images, direct modeling using a generative Normalizing Flows approach can be computationally expensive and may lead to suboptimal performance.

To address this, this work employs Principal Component analysis, and trains a Real NVP model [2] on the data containing approximately 90% of the variance. We'll then use this model to observe how interpolating between the latents of different classes affects the image generation.

2 Methodology

The Olivetti Faces dataset will be downloaded from sci-kit learn. After that, we will apply a PCA [3] selecting the components that explain 90% of the variance. This turned out to be 66 variables. Our model works best with a power of two size of latent, so we'll round it to 64 (which explains 89.70% of the variance).

It's interesting to see which pixels influence the most on the PCA (have higher importance). We can plot the importance per pixel in the first 10 components 1

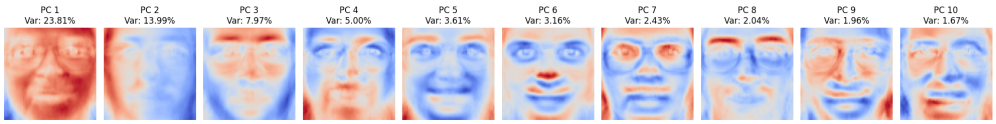

Figure 1: Pixel importance for each of the first 10 components of the PCA

The first component focuses a lot on the pixels of the face, especially the right side. The second one on the left-side background (and a bit the left side of the face). Third one and fourth are focused on the top and the bottom of the image respectively, highlighting zones like the eyebrows or the mouth. Component 7 is interesting because we can observe an outline of some glasses having a negative weight on the PCA (which would indicate that darker pixels on that region contribute to the variance explained by that component).

After that, we will train a Normalizing flow on this PCA-transformed data, using the normflows library. We have decided to use 14 coupling layers.

The baseline model has been trained with 5000 epochs, but we also studied the effects of training for less (comparing the log_probs of the training dataset).

With a trained model, we can take two images from the training data (from different classes), transform them into latent, interpolate between those latent vectors and pass those through our Real NVP to observe which images does it generate.

We can observe the loss for training a model with K=14 for 5000 epochs on 2.

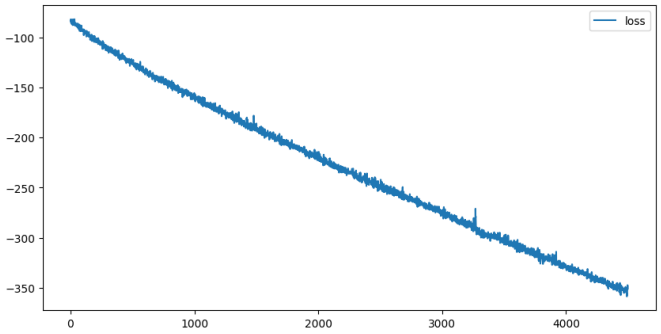

Figure 2: Loss curve of Real NVP training for 5000 epochs

3 Results

We can observe are some samples generated by the model trained with 5000 epochs 3, and also with fewer.

Figure 3: Samples 5000 epochs

Figure 4: Samples 2500 epochs

Figure 5: Samples 1000 epochs

Figure 6: Samples 500 epochs

Figure 7: Samples 100 epochs

As expected, they all look like faces from the dataset. Some are pretty close to real faces, while others (like the bottom right) have some features that are a bit blurry, especially the nose, mouth, and mustache. An interesting finding is that the model, when generating samples, doesn't improve too much even if we train it for 10 times more. We'll later see that the log probabilities improve, but in the examples, the samples from the model trained for 500 epochs aren't far away from the ones by the model trained for 5000.

After this, we can evaluate the log probability of our data. In table 1 we can observe the results for each of the first 20 classes.

Class	Log Probability
0	20740.2363
1	19319.7949
2	20124.3633
3	19761.5566
4	22512.5957
5	20991.8730
6	20049.3633
7	18992.2793
8	21534.7148
9	19742.7852
10	19672.6797
11	19120.8555
12	20119.4980
13	20684.0781
14	20796.5176
15	19848.6914
16	19320.4219
17	19496.9023
18	17937.5742
19	17903.4336

Table 1: Log Probabilities of Different Classes (only first 20)

The complete LogProbs of the whole training data is 122229.4765625. This number is pretty high and indicates that the RealNVP model is able to capture the distribution of the dataset. It's interesting to see that if we train with a subset of the dataset and try to evaluate the logprobs on the other samples, the resulting values are very low, very often getting -inf values and nan (log(0)). To avoid having these issues, we had to drastically reduce the K (to 2) and the training epochs to 50.

We can also observe how some classes have higher probabilities than others. For example, class 4 has a very high log probability, which means the model is better at that one compared to class 19, which only has 17903. Since the classes are balanced, that could also indicate that class 4 contains images of someone who has very common features, while 19 is more unique.

As an experiment, we decided to train the same model (K=14) for fewer epochs. We evaluated the log_probabilities on the whole dataset (like before), and obtained the following graphic 8.

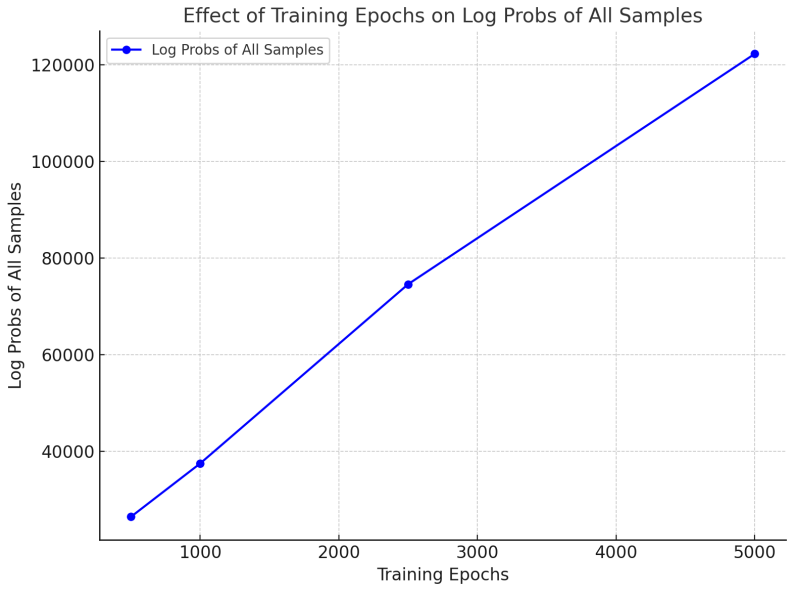

Figure 8: Effect of training epochs on the log-probs of all samples

As we can observe, the relationship between the number of epochs and the log_probs is linear: the more training the model does, the higher the probabilities it assigns to the dataset samples. Probably, with more epochs we would reach a point where this relationship is no longer the same (we observed that the training loss of the model still hadn't reached a plateau with 5000 epochs. A smaller model's (for example, K=8) loss more or less converged, or at least started to flatten, with that number of epochs).

Having trained our model, we were able to select some of the classes and interpolate between them in the latent space. With the Real NVP model and the inverse of the PCA, we were able to reconstruct those latents into images, as shown on 9

Figure 9: Examples of interpolation between different classes

The results are pretty interesting and pretty good. The interpolation creates new faces, that take and combine features from the two. For example, we can see the effects of interpolating between a face without glasses and one with, where the glasses slowly start to appear in the image. A similar thing happens for faces with a beard. It seems that the model struggles a bit with mustaches: in some interpolations, it transforms the mustache into a mouth (and vice-versa), which produces some interpolations that are a bit weird.

We can also observe the results of interpolating examples of the same class 10

Figure 10: Examples of interpolation between the same class

In this case, we selected images that had different viewing angles of the faces. We can observe how the interpolation can more or less generate faces that correspond to the intermediate angles.

Basically, these results show us that the latent space encodes some meaning: it contains information about the position of the face, the brightness, features it contains... Moving along that space, from sample to sample, the model is able to generate new faces that are a combination of those features. Theoretically, this could allow us to condition the model on some characteristics.

Another thing we tried was to interpolate to an image that didn't exist in the original dataset. To do so, we took a photo, rescaled it to 64x64 pixels, and transformed to a grayscale image (as similar to the existing faces as possible). After that, we used the PCA and the model from before to obtain a latent of this new image, and interpolated as before. This also allowed us to see how well the model could reconstruct an image it had not seen.

As seen before with the log-probs, with the model trained for 5000 epochs we weren't able to reconstruct an image that it hadn't seen, since it was too focused on the distribution of the existing images. However, with a much weaker model, we were able to get it to generate a latent from this new image (that didn't contain nan values) and reconstruct from there. We also tested this interpolation with an image containing the mean values of all the faces, and the mean latent vector.

(a) New face

(b) Interpolation / reconstruction results with the new face

Figure 11

(a) Mean of all faces

(b) Interpolation / reconstruction results with the mean face

Figure 12

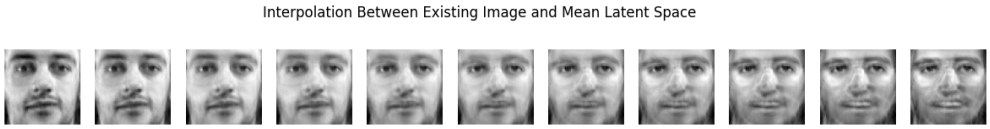

Figure 13: Interpolation to the mean latent vector of all faces in the Olivetti dataset

The model can reconstruct the new face into one that is a bit similar (especially in the lighting, since the new image had a clear difference between the left and right side of the face, and the reconstructed one also has that effect). In the process of interpolating to the mean face, there seems to appear a different face between the reference one and the mean of all faces that isn't as blurry as this final one but is also different from the starting one. Finally, we see that the mean latent vector produces a really weird face, which seems to have at the same time many of the features: showing the teeth, glasses, long hair, beard...

4 Conclusions

After seeing the results, we can extract some conclusions. For starters, we were able to reduce the dimensionality of the images from 4096 components to only 64 and the results seem to show that PCA was very effective. By retaining almost 90% of the variance in 64 variables, our model became more computationally light but was still able to reconstruct images very similar to those in the dataset.

The Real NVP was very good at capturing the distribution of the dataset: the log probabilities of the training data were high, and the loss curves indicated that with further training those could increase even more. However, we also saw that really the generated samples weren't that much better compared to lower log-probabilities models (trained with fewer epochs, or different K). Some exhibited blurred noses, inconsistent mouths, and transparent glasses.

Interpolation experiments showed that by transitioning between latent representations, the model was able to successfully merge diverse features, making them appear or disappear gradually. This shows that those latent spaces

contain information about the different traits and features, and one could "modify" some generations by moving alongside some directions on that latent space, similar to what researchers at OpenAI did with [4]. Interpolating for the same class showed that some of the encoded information on the latent space also corresponded to the facial angle.

We also saw that the model overfitted a lot: calculating the log-probabilities of unseen examples resulted in nan or infinite values, and we had to reduce a lot the model and training epochs to be able to somewhat generalize.

In conclusion, this work showed the strenghts of Real NVP and flow models in learning and generating realistic images, combined with PCA for dimensionality reduction. They demonstrated the hability to interpolate between samples, but lacked a bit in fine-grained details and when generalizing to unseen data (even if that data was, in theory, from a similar distribution).

Suggestions for further work could include trying more advanced models, observing the effects of selecting more or less PCA components, or analyzing better the latent space of the model to identify more precisely the different features.

References

[1] C. L. N. U. of Cambridge, "Olivetti faces dataset," 1992. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_olivetti_faces.html.

[2] L. Dinh, J. Sohl-Dickstein, and Y. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2017.

[3] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[4] D. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *NeurIPS*, 2018.