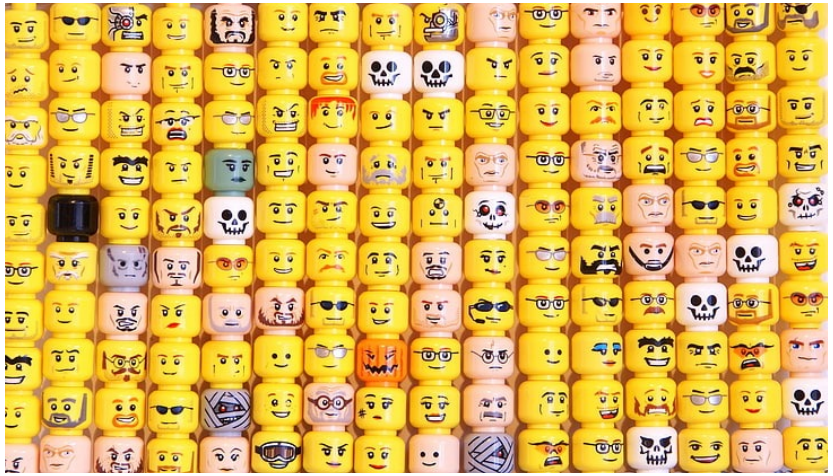
Variational Autoencoder for Facial Image Generation with Bayesian Optimization

Probabilistic Machine Learning

Annalisa Paladino

Introduction

Dataset: lego facial images



Goal: to achieve high-quality image generation

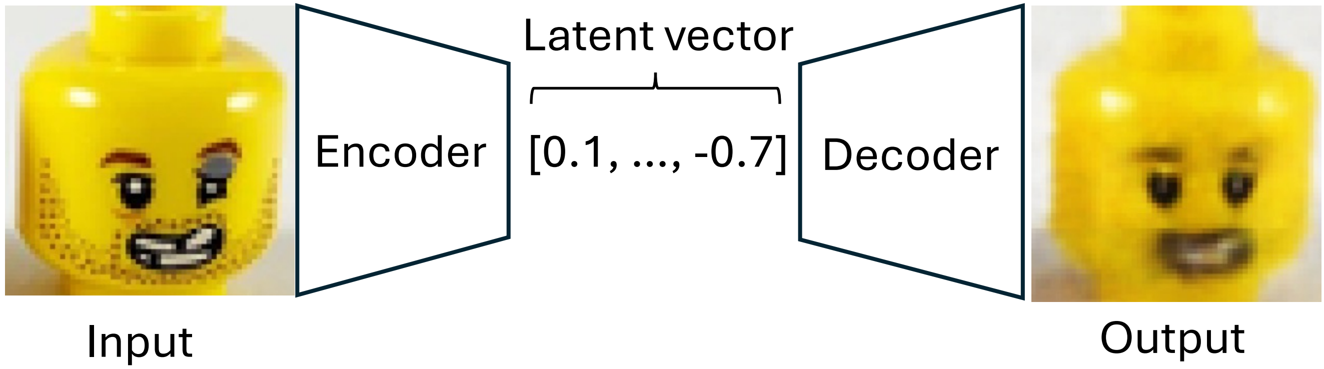
How?

- **VAE** to generate facial image
- **Bayesian optimization** to optimize VAE's hyperparameters

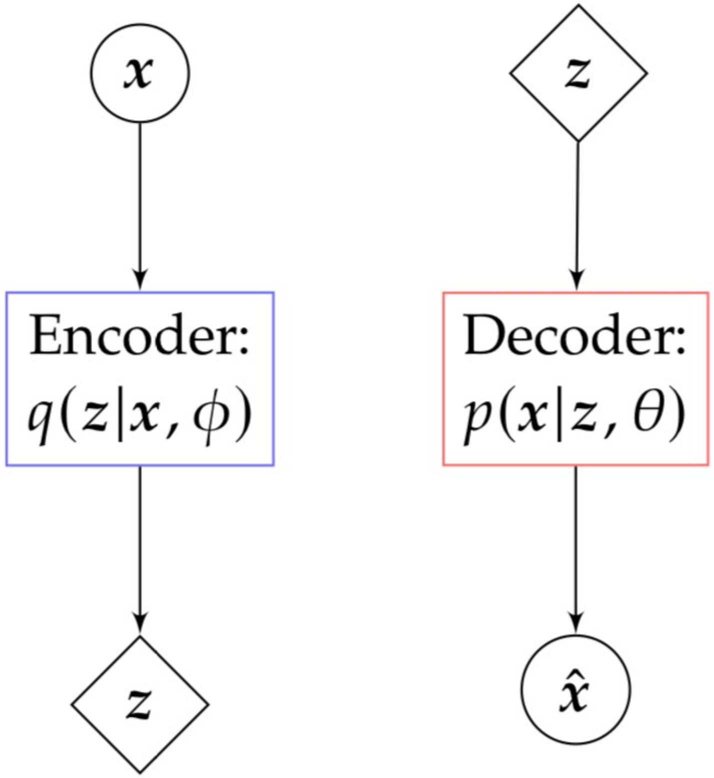
Variational Autoencoder (VAE)

ENCODER: takes as input an image and compress it into a compact, **latent** representation. The VAE outputs the parameters that define a *probability distribution* (mean and log-variance) over the latent space

DECODER: samples a point from the distribution in the latent space and attempts to **reconstruct** the original input as accurately as possible



Variational Autoencoder (VAE)



The input x is mapped through the encoder into a latent space z from which we can reconstruct $\hat{x}$ via the decoder

Goal: $\hat{x}$ is as **similar** as possible to x

$$p(x, z|\theta) = p(x|z, \theta)p(z)$$

$$q(z|x, \phi) = N(\mu_z(x, \phi), \sigma_z^2(x, \phi) \cdot I)$$

Reparametrization trick

Decompose a random variable z from the distribution into a **deterministic component** and a **stochastic component**

1. **Introduce a new independent random variable ε:** ε serves as a source of randomness entirely separate from the sampling process within the VAE

2. **Reparameterize the latent variable:** create a scaled version of the new random variable (ε) using the standard deviation predicted by the encoder

$$z = \mu + \sigma \cdot \varepsilon$$

source of randomness

deterministic outputs
from the encoder

Loss = MSE + KLD

Mean Squared Error

Measures the **reconstruction error** between the original image and the reconstructed image

Kullback-Leibler Divergence

Measures the **difference** between the **learned latent distribution** and a **standard normal** distribution, encouraging the model to learn a regularized latent space

Bayesian Optimization

Goal: find the set of input parameters x that **maximize** a function $f(x)$

$$\underset{x \in A}{\operatorname{argmax}} f(x)$$

In our case:

- The input of f is a vector of hyperparameters, from a subset of $\mathbb{N}^2$
- The output is the validation loss $\mathbb{R}^+$

Bayesian Optimization - *properties*

- A is a set of points x, which rely upon less (or equal to) than 20 dimensions ($\mathbb{R}^d, d \leq 20$)

- The objective function f is **continuous**

- f is **expensive** to evaluate due to its computational cost

Bayesian Optimization - *properties*

- f is a **black-box** : it lacks structure like concavity or linearity
- f is **not differentiable**: we observe only $f(x)$ and no first- or second- order derivatives
- Our focus is on finding **global optimum** → the goal is to have a guide when exploring the hypermater’s space

Bayesian Optimization - *components*

Surrogate model: a statistical model to **approximate** the objective function $f(x)$

Acquisition function: function that **guides** where to sample next

Algorithm

Place a Gaussian process prior on f

Observe f at n_0 points according to an initial space-filling experimental design. Set $n = n_0$

While $n \leq N$ **do**

> Update the posterior probability distribution on f using all available data
>
> Let x_n be a maximizer of the acquisition function over x, where the acquisition function is computed using the current posterior distribution
>
> Observe $y_n = f(x_n)$
>
> Increment n

End while

Return the point evaluated with the largest $f(x)$

Surrogate model - Gaussian Process

A **Gaussian Process** (GP) is a stochastic process indexed by a continuous variable $x \in \mathbb{R}^n$, such that all finite dimensional distributions are multivariate Gaussian

GP is uniquely defined by:

- μ: $\mathbb{R}^n \to \mathbb{R}$ function that computes the **mean** at every point x
- K: $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ function that computes the **covariance** between any two points

Surrogate model - Gaussian Process

$$f \sim GP(\mu, K) \Leftrightarrow \forall x_1, \dots, x_n \in \mathbb{R}^n, p(f) = p\big(f(x_1), \dots, f(x_N)\big) = N(\underline{\mu}, K)$$

With:

$\underline{\mu} = \big(\mu(x_1), \dots, \mu(x_N)\big)$

$K = \big(K_{ij}\big)$ with $K_{ij} = k(x_i, x_j)$

Let's fix the mean to 0 and the kernel to be the Matern with parameters

$$\rho = 1, \qquad \nu = 2.5$$

Acquisition function - Expected Improvement

Improvement: how much better the function value $f(x)$ at a given point x is compared to the best known function value f_n^* achieved at step n

$$[f(x) - f_n^*]^+$$

Where

$a^+ = \max(a, 0)$

$f_n^* = \max_{m \le n} f(x_m)$

But... $f(x)$ is **unknown** ➡ **EXPECTED IMPROVEMENT**

Acquisition function - Expected Improvement

Given a surrogate model (Gaussian Process) that provides a **mean** and **standard deviation** of the predicted function value at point x, the **expected improvement** is

$$\mathrm{EI}_n = \mathbb{E}[(f(x) - f_n^*)^+]$$

Where

$a^+ = \max(a, 0)$

$f_n^* = \max_{m \leq n} f(x_m)$

Hyperparameters optimized

- **Hidden dimension**: Controls the number of neurons in the fully connected layers of the model

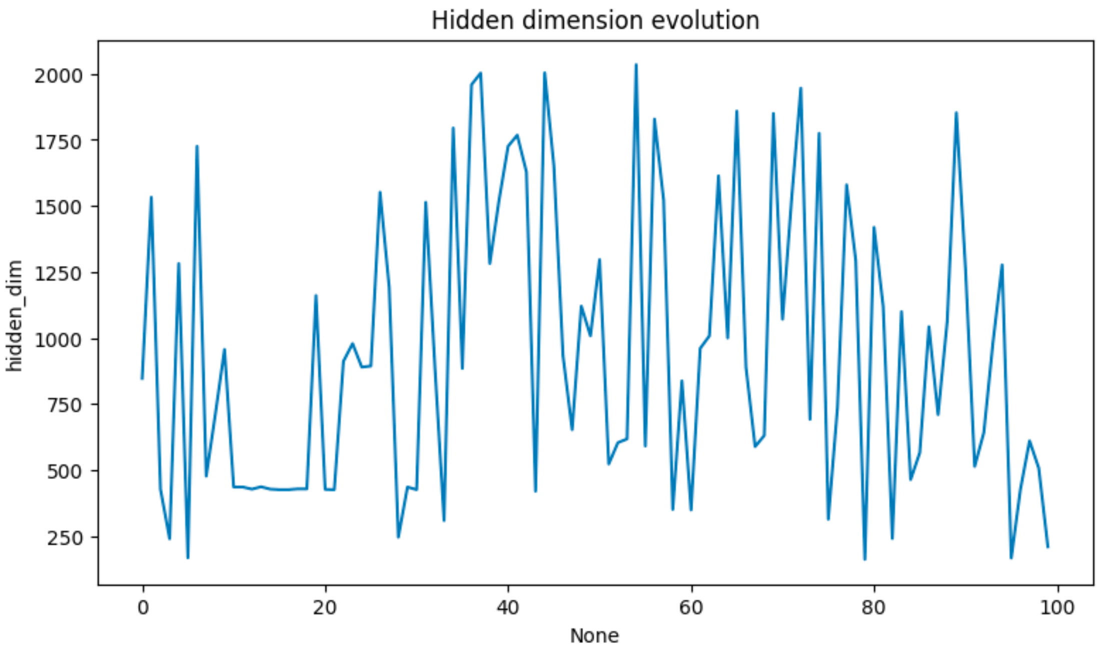
- **Latent dimension**: Controls the size of the latent space, which impacts the model's ability to generate varied outputs

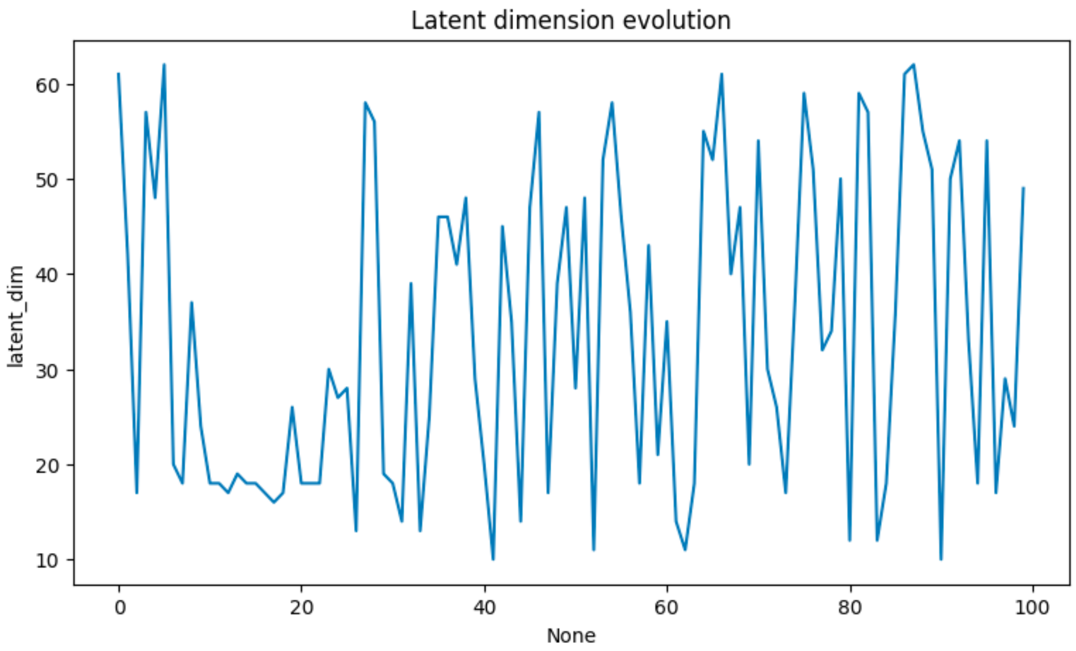
Bayesian Optimization samples the hyperparameters over:

- 10 random points to explore the hyperparameter space
- 90 further iterations to refine the search around promising hyperparameters

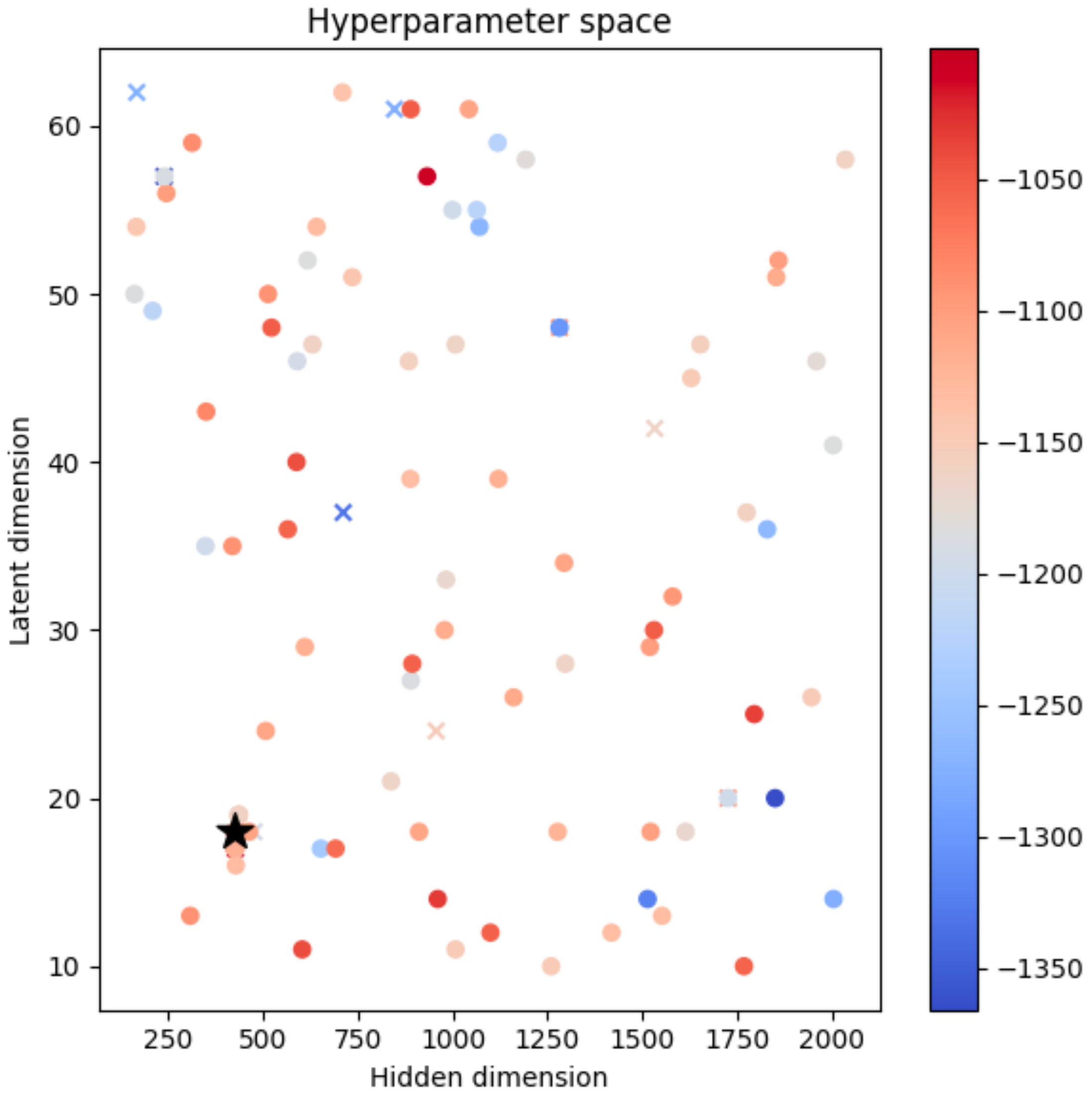
Epochs: 50

Hyperparameters evolution





Hyperparameter space



Experimental set-up

The optimization process identified the best hyperparameters:

Hidden dimension: Optimal size found within the range of 128 to 2048

Latent dimension: Optimal size found between 8 and 64

The VAE model is instantiated with these values!

Experimental set-up

Optimizer: Adam

Epochs: 400

Initial learning rate: 1e-3

Final learning rate: 5e-7

Learning rate scheduling: to decay the learning rate over time

Gradient clipping: to avoid instability and to prevent exploding gradients

clip_grad_value: 1.0

Batch size: 64

Loss

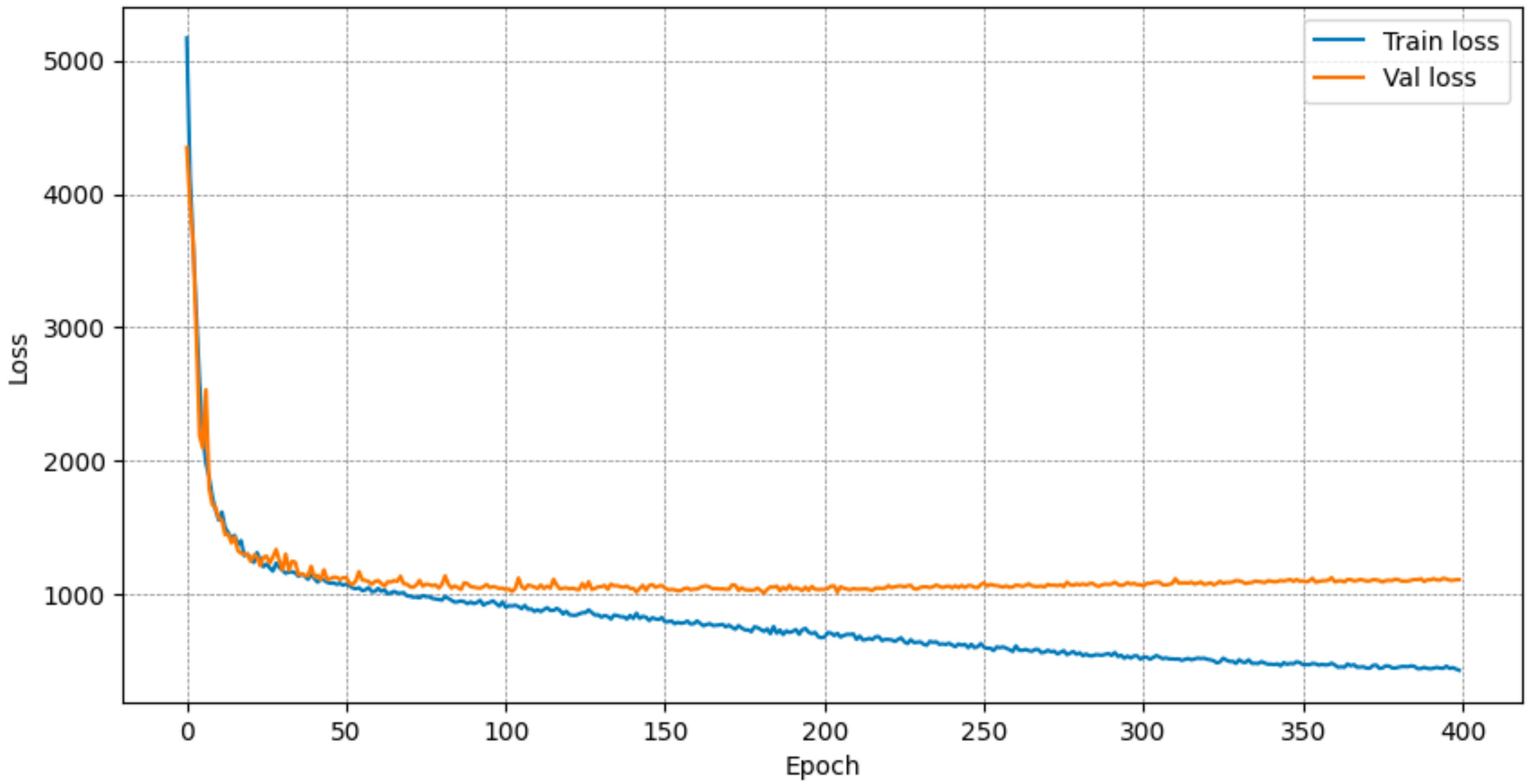

Train loss
Val loss
5000
4000
3000
2000
1000
Loss
0
50
100
150
200
250
300
350
400
Epoch

Image Generation

Face Morphing

The latent space is **continuous** and **smooth**

We can **interpolate** between points in this space to generate smoothly transitioning variations of our data

Conclusions

The model learned to encode images into a latent space and generate reconstructions

Bayesian optimization tuned the model’s hyperparameters, improving the model’s performance without requiring extensive manual tuning

The model demonstrated smooth transitions between facial images through interpolation


Thank you for your attention !