Feature alignment as a generative process

Tiago de Souza Farias* and Jonas Maziero

Departament of Physics, Center for Natural and Exact Sciences, Federal University of Santa Maria,
Santa Maria, Brazil

Reversibility in artificial neural networks allows us to retrieve the input given an
output. We present feature alignment, a method for approximating reversibility
in arbitrary neural networks. We train a network by minimizing the distance
between the output of a data point and the random output with respect to a
random input. We applied the technique to the MNIST, CIFAR-10, CelebA, and
STL-10 image datasets. We demonstrate that this method can roughly recover
images from just their latent representation without the need of a decoder.
By utilizing the formulation of variational autoencoders, we demonstrate that
it is possible to produce new images that are statistically comparable to the
training data. Furthermore, we demonstrate that the quality of the images can
be improved by coupling a generator and a discriminator together. In addition,
we show how this method, with a few minor modifications, can be used to
train networks locally, which has the potential to save computational memory
resources.

1. Introduction

Feature visualization (Olah et al., 2017) is a set of techniques for neural networks
aiming to find inputs that maximize the activation of one or more selected neurons
from the same network. Usually, feature visualization is used as a method for model
interpretability, where one seeks to understand a neural network by analyzing how much
each neuron contributes to a neural network by perceiving the images generated by
these techniques. The process of obtaining these inputs is, in a sense, an attempt toward
reversing a neural network. Since a neural network is composed by functions that map
inputs to outputs, the visual representation of a feature is the input we would have given
a target activation for a group of posterior selected neurons.

The reversibility of neural networks relates to how well one can reverse the map from
the activation of target neurons back to the input neurons (Gomez et al., 2017). In most
cases, neural networks are not reversible, primarily due to three reasons: (1) The presence
of non-reversible activation functions [e.g., ReLU (Nair and Hinton, 2010)], which means
that in general, it is impossible to directly recover the input value x given the output
value $f(x)$. (2) Non-orthogonal weights, as there are neither constraints nor incentives
for their matrix representation to converge to having this property. (3) Lack of one-to-
one relationships as a result to the reduction of dimension as the information is passed
through each layer of a network. In addition to the reversing mapping, reversible neural

networks have the benefit of memory efficiency: unlike non-invertible neural networks, which must store all of the activations for the backward pass during training, reversible neural networks only need to store a portion of the activations in order to update the trainable parameters.

Reversibility also constrains the number of possible models, as many possible parameters configurations model the data. For example, if one considers an analytical function that one wants a sufficiently parameterized neural network to approximate, with the pair of data $\{x, f(x)\}$, several local minima estimate the function $x \rightarrow f(x)$, each of which was obtained by a different random initialization of the neural network parameters (assuming optimal convergence). By restricting the reversibility $f(x) \rightarrow x$, we can reduce the number of optimal points toward which a neural network can converge. Since many local optima converge to comparable losses, local optima do not pose a problem for neural networks; however, they lack interpretability because the inputs cannot be recovered from a given output.

Memory is often a bottleneck for neural networks. Modern deep learning techniques frequently use the backpropagation algorithm (Linnainmaa, 1976; Rumelhart et al., 1986), which requires the storage of all network activations in order to update its parameters. Local training rules enable a more effective memory optimization of neural networks (Baldi and Sadowski, 2016). By constraining the trainable parameters, such as the weights, to be updated only by local variables (the information contained in the neurons that share the same parameter), we can reduce the memory requirements to load a model in hardware such as CPUs and GPUs. This constraint can conserve memory resources and has a wide range of potential applications, including low-memory devices (Sohoni et al., 2019; Velichko, 2020), training large batch sizes (You et al., 2017; Gao and Zhong, 2020), and, even training very large neural networks (Jing and Xu, 2019).

Our goal in this paper is to show that feature alignment can be used for approximate reversibility of neural networks as well as sampling of images. This approximation is based on performing gradient descent on the input space while simultaneously training a network to estimate the input given an output. To show the feasibility of the proposed technique, we make use of generative networks to generate samples statistically similar to the training data by making use of approximated reversibility. We also adapt the technique for local training, showing that is possible to reverse an encoder by mapping the output latent vector back to the images of a dataset with only local variables.

2. Related work

Several works have been done in the area of feature extraction, especially applied for model interpretability and explainability (Gilpin et al., 2019; Fan et al., 2021; Ismail et al., 2021; Shahroudnejad, 2021; Thakur and Han, 2021). These techniques, used for extracting features, usually consist in activation maximization (Mahendran and Vedaldi, 2016; Ellis et al., 2021), where a group of neurons, which can involve from a single neuron up to an entire layer (or channel for convolutions), is selected to extract the feature by maximizing its activation. Many of these techniques of feature extraction consist in studying features in already pre-trained classifiers (Nguyen et al., 2016b). Other techniques consist in searching for features in the latent space (Shen et al., 2020). Feature extraction can also be utilized for understanding which parts of an input contribute the most for the target activations (Springenberg et al., 2015; Zintgraf et al., 2017; Selvaraju et al., 2020).

In a generative process, we want to produce new examples with the same statistical distribution as the training data. There are several different techniques to model the data for a generation. Among these techniques, autoencoder based networks, generative adversarial networks, and normalizing flows are very popular. Autoencoders (AE), while not generative networks, they constitute of building blocks for other generative networks and offer insights about mapping the input to other representations. Autoencoder consist of two networks: an encoder that projects the inputs into a vector, usually with a smaller dimension, and a decoder that reconstructs the input from this vector. The compressed vector has a high-level representation of the model, in which each neuron contributes to properties beyond the data level at the input layer (Lee et al., 2011). Autoencoders are commonly trained in an unsupervised fashion, nevertheless, some variants include labeled information to further increase training for a specific objective. Variational autoencoders (VAE) (Kingma and Welling, 2014, 2019; Doersch, 2021) gives autoencoders generative capability by projecting the data into a probabilistic latent vector, thus we can generate data statistically similar to the training data by sampling random latent vectors and projecting them to a decoder network. Generative adversarial networks (GANs) (Goodfellow et al., 2014; Gui et al., 2020; Salehi et al., 2020) are another example of a generative method. By having two competing networks, a generative network which takes a random low-dimensional input and outputs an image, and a discriminator network that compares the images from the training dataset and the sampled ones from the generator. The competition arises by training the generator to fool the discriminator by generating images as closest to the training dataset as possible. Normalizing flows (Kingma and Dhariwal, 2018; Papamakarios, 2019; Kobyzev et al., 2020) is another generative paradigm that generate images by transforming a simple distribution to a more complex one by a series of reversible transformations.

Diffusion models is another method that can be utilized to generate samples that are statically comparable to a dataset (Sohl-Dickstein et al., 2015; Ho et al., 2020; Rombach et al., 2022). They are trained to predict the noise that is presented in a

sample by repeatedly exposing them to increasing levels of noise as part of their training. The state-of-the-art capability of this method, which can generate samples with high fidelity and that are similar to the training dataset, is the primary benefit of using this method. On the other hand, diffusion models take a long time to sample because they require a large number of steps to denoise an image, which is a process that is very computationally intensive.

There have been works combining autoencoders with GANs (Larsen et al., 2016). The work done in Dosovitskiy and Brox (2016) and Nguyen et al. (2016a) is related to ours. They synthesized new images with the same statistics as the training data by inputting features to a generator network. The main difference is that, in these previous articles, the features are obtained with a pretrained network.

Most works on reversibility consist in architectural changes of neural networks (Baird et al., 2005; Grathwohl et al., 2018; Schirrmeister et al., 2018; Atapattu and Rekabdar, 2019; Behrmann et al., 2019). These changes guarantee a one-to-one relationship between inputs and outputs. BiGAN (Donahue et al., 2017) constructs a generative network and a reverse network that inputs images back to noise, which can be used to obtain a latent representation of a dataset directly. Dong et al. (2021) shows that is possible to reverse neural networks in the case of reconstruction of images.

Local learning rules have been explored since Donald Hebb proposed a simple model for learning in the brain (Hebb, 1949). The main advantage of this kind of learning algorithm is requiring lower memory resources. Some works are biologically inspired (Krotov and Hopfield, 2019; Lindsey and Litwin-Kumar, 2020), while others focus solely on efficiency (Isomura and Toyoizumi, 2016, 2018; Wang et al., 2021; Guo et al., 2022). There is a growing body of work discussing whether the brain does backpropagation (Whittington and Bogacz, 2019; Song et al., 2020), with some approximations for training artificial neural networks (Lillicrap et al., 2020; Millidge et al., 2020; Laskin et al., 2021; Salvatori et al., 2021).

Another approach for saving memory resources is gradient checkpoint (Dauvergne and Hascoët, 2006; Chen et al., 2016; Kumar et al., 2019; Sohoni et al., 2019), where memory is traded with computation time by re-evaluating neurons when they are needed for backpropagation instead of storing their activations all at once. While this technique decreases the amount of memory necessary to train a neural network, it requires many forward propagation calculations on the network, depending on its size, which can increase time consumption, while local learning rules, as opposite, require only one forward propagation to update the parameters.

Table 1 summarizes the related works. We classified each method by four properties: being able to train with new data, whether the method can reconstruct data from a latent space, if the method is able to produce new samples, and reversibility. Three methods have all four properties:

normalizing flows, diffusion models and feature alignment. Architecturally, normalizing flows is composed entirely by reversible layers, while feature alignment allows for arbitrary networks. Diffusion models contain a reversibility restriction that is reversing samples from noise; consequently, diffusion can only reverse samples with higher noise to lower noise. This requirement does not apply to feature alignment because the mapping can be done with any number of dimensions from the input space to the output space and loss does not always require noise for optimization.

3. Methods

The method of feature alignment is covered in this section. It consists of two phases: first, we perform a gradient descent on a random input using a loss function that measures the distance between the encoded random input and an encoded image. After that, the network is then trained on a different loss function, which evaluates the distance between the inputs and the gradient that was performed on it. By doing this, the network is able to learn the inverse map that leads from its outputs to the inputs that correspond to those outputs, thus recovering the information that triggered its activation.

3.1. Feature alignment

The feature alignment encoder consists of a encoder with parameters θ and an arbitrary number of latent variables as the output. From a dataset $\mathbf{x} \in \mathbf{X}$, $\mathbf{z}_x = E(\mathbf{x}; \theta)$ is the output from an input $\mathbf{x}$. With the same network, $z_r = E(\mathbf{r}; \theta)$ is the output from a random input $\mathbf{r}$, chosen from some probability distribution, with the same dimension as the input data.

The feature $\hat{\mathbf{r}}$ of $\mathbf{z}_x$ is obtained by minimizing a distance function $\mathcal{L}(\mathbf{z}_x, \mathbf{z}_r)$ with respect to the random inputs $\mathbf{r}$. We choose a gradient flow for minimizing this distance, since it can evolve the random input continually, as follows in the Equation (1).

$$\frac{\partial r}{\partial t} = -\frac{\partial \mathcal{L}}{\partial r}. \tag{1}$$

Since $\mathbf{z}_x$ is fixed, $\mathbf{r}$ will evolve such as the function of the random variables will approximate $\mathbf{z}(\mathbf{x})$ as much as possible (see Appendix A). We want to solve Equation (1) as efficiently as possible in time and memory. By discretizing the gradient flow, we obtain an approximation for the feature, as shown in Equation (2).

$$r^t = r^{t-1} - \tau \frac{\partial \mathcal{L}}{\partial r}, \tag{2}$$

with τ being a hyperparameter that weights the contribution of the gradient. Equation (2) is similar to activation maximization, except that we are minimizing for the neurons to have a target

TABLE 1 Comparative table among previous works from the literature and with this article.

Method	Trainable	Reconstruction	Sampling	Reversible	References
AE	✓	✓	✗	✗	Lee et al., 2011
VAE	✓	✓	✓	✗	Kingma and Welling, 2014
GAN	✓	✗	✓	✗	Goodfellow et al., 2014
BiGAN	✓	✓	✓	✗	Donahue et al., 2017
Feature extraction	✗	✗	✗	✓	Mahendran and Vedaldi, 2016
Normalizing flow	✓	✓	✓	✓	Kingma and Dhariwal, 2018
VAE-GAN	✓	✓	✓	✗	Larsen et al., 2016
Reversible guidance	✗	✓	✗	✓	Atapattu and Rekabdar, 2019
Diffusion models	✓	✓	✓	✓	Rombach et al., 2022
Feature alignment	✓	✓	✓	✓	This work

FIGURE 1

An encoder that has been trained using feature alignment for the purposes of reconstruction. **Left**: the encoder receives an input that is batched from training data as well as a random input, which is updated by minimizing the distance between their latent representations. The network is trained to approximate $\hat{r} \approx x$. **Right**: during inference, we reconstruct x by only using its latent representation z_r and doing gradient descent on the random input r.

This second loss function is used for training the encoder by optimizing its parameters. As the neural network is trained, the encoder learns, not only to map the inputs to the latent variables, but also the reconstruction of the inputs from the latent vector. Following training, we can reconstruct the inputs by knowing only the latent vector. Figure 1 and Algorithm 1 summarize the feature alignment technique. First, using an encoder $E(\theta)$, we obtain the latent representation z_x of an input, x, which can be an image. The same encoder is then given a random input of the same size as the input that is drawn from a predetermined distribution (such as a uniform or Gaussian distribution), outputting the latent representation z_r. We then perform a gradient descent on the random input for T steps (a chosen hyper-parameter). Once we have both latent representations, we update the encoder parameters θ by minimizing the distance between the optimized random input $\hat{r}$ and the input x.

activation, which is the latent representation of an image x. These updates are done in T time steps. Properly optimized, the solution to Equation (2) converges to the input x by approximating the inverse of the weights (see Appendix A). So, by optimizing the parameters of the network, the weight matrix between layers will have the orthogonal property $\mathbf{w}^T\mathbf{w} = \mathbf{I}$, which implies in approximated reversibility (see Appendix B).

After we extract the representation $\hat{\mathbf{r}}$, we measure how similar it is to the inputs $\mathbf{x}$ by a new loss function $\mathcal{C}(\mathbf{x}, \hat{\mathbf{r}})$.

3.2. A toy example

To gain a better understanding on how feature alignment works, here we will look at a straightforward example. Let's say we want to approximate a function $y = f(x)$ with a neural network. While we can approximate y with a sufficiently parameterized neural network $\mathcal{F}(x; \theta)$, we can not recover x given only $\mathcal{F}(x; \theta)$ for functions that do not have a one-to-one relationship.

```
z_x = E(x; θ)
initialize r = x.shape from a random distribution
t = 0
while t < T do
    z_r = E(r; θ)
    L = ||z_x − z_r||²₂
    r = r − τ ∂C/∂r
    t = t + 1
end while
r̂ = r
C = ||x − r̂||²₂
update θ by optimizing C
```

Algorithm 1. Training with feature alignment.

With the feature alignment method, we can constrain the network $\mathcal{F}$ to be able to approximate the reversible map $x = f^{-1}(y)$. In this example, we will look at the function $y = \sin(3\pi x)$. The network used consists of two fully connected hidden layers with 1,024 neurons each, with both input and output single neurons. The network trained with FA has an extra neuron in the output layer to act as a latent variable because $\sin(3\pi x)$ has a correspondence many-to-one. The equation

$$\mathcal{L} = ||l_x − l_r||_2^2 + \alpha ||f_x − f_r||_2^2, \qquad (3)$$

represents the auxiliary loss function for reversing the network, with a hyper-parameter α, which in this example was set to $\alpha = 0.01$, f_x is the output of the network we want to approximate and f_r is the output given a random input, l_x is the latent neuron and l_r is the latent neuron from the random input. Then, we can recover x by using the equation $\hat{r} = r − \frac{\partial \mathcal{L}}{\partial r}$.

The parameters of network are updated to minimize both the difference between the input to its approximation and the output to the function we want to approximate, with the loss function:

$$\mathcal{C} = ||y − f_x||_2^2 + ||x − \hat{r}||_2^2. \qquad (4)$$

Figure 2 shows the results for a network trained with and without feature alignment. We can observe that in the absence of FA, the network is only able to approximate the inputs partially. On the other hand, when the loss of feature alignment is included, we have complete approximation within the entire function domain.

3.3. Variational autoencoders with feature alignment

In the context of generative processes, autoencoders, in general, are unable to generate new samples with the same statistical distribution as the training data. The latent variables from the data, if associated with a distribution of variables, may be too complicated or convoluted for effective sampling. To enable feature alignment with sampling, we use the variational autoencoder (VAE) formulation, without a decoder network. As a result, the inverse of the encoder becomes its own decoder, just as was previously with autoencoders. In the VAE, the output of the encoder is coupled with two layers that return the mean value μ_x and variance σ_x^2 of the data. We constrain the latent vector to have a distribution that is easy to sample (typically a Gaussian distribution), by comparing two probability distributions using a metric such as the Kullback-Leibler divergence. Subsequently, the cost function in Equation (5) is used to train the feature encoder with a constraint to the output latent variables from a known random probability distribution $p(\mathbf{z})$, from which we can easily sample. The constant β is a hyper-parameter that improves the disentanglement representation of the data by regularizing the latent vector (Higgins et al., 2016; Burgess et al., 2018; Sikka et al., 2019):
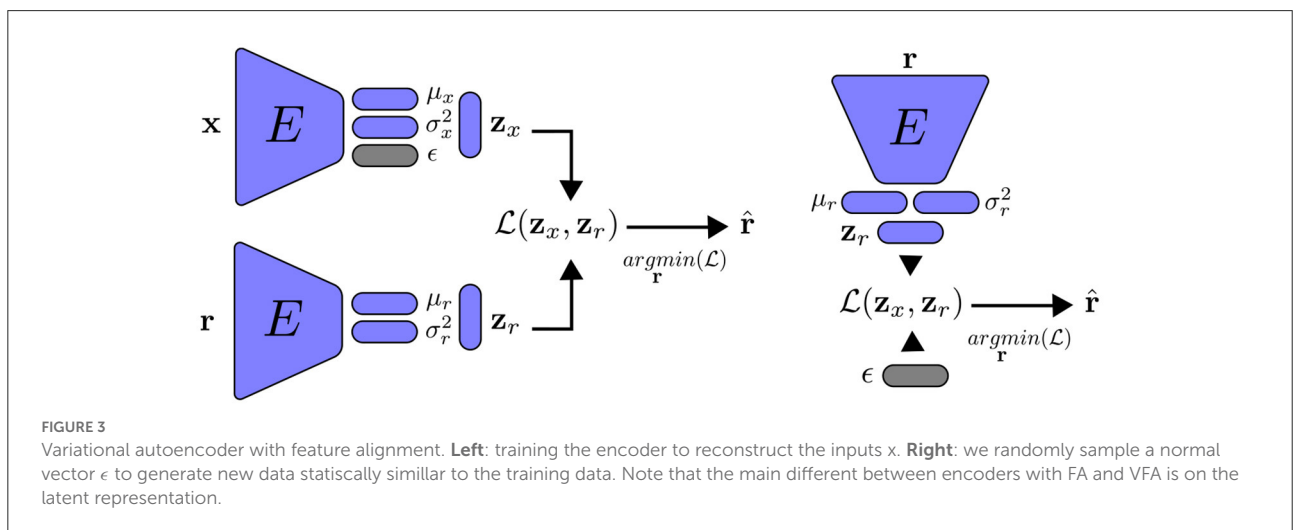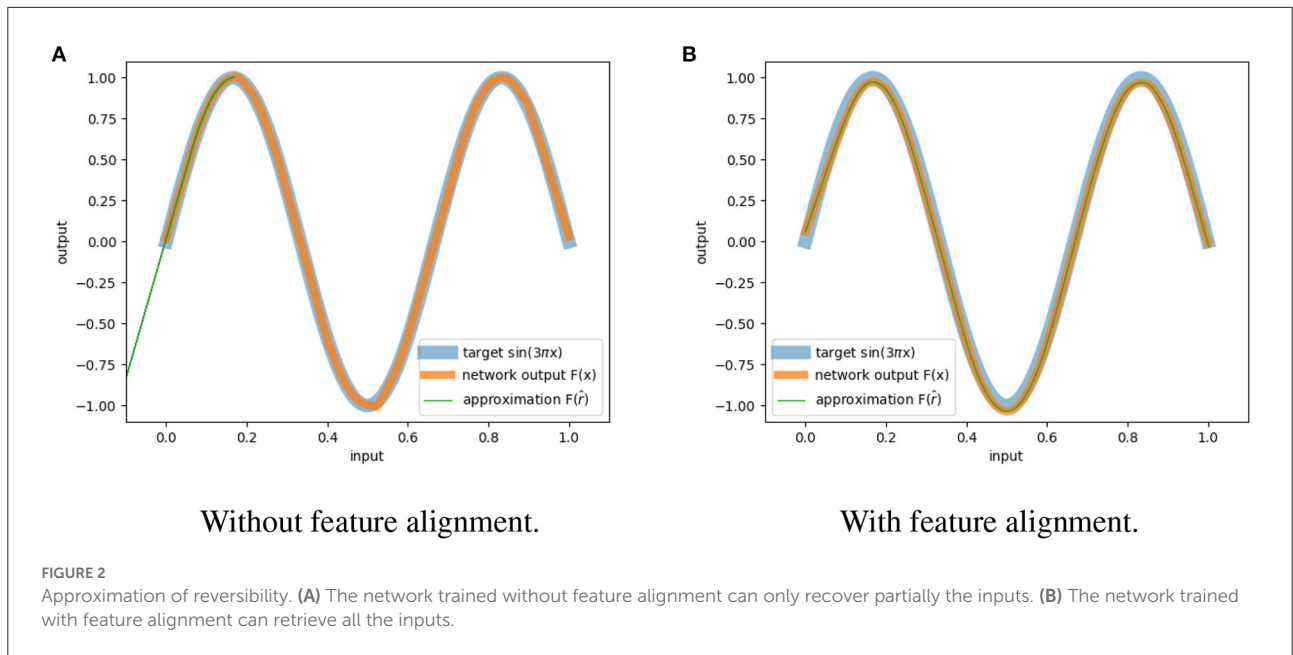
$$\mathcal{L} = ||\mathbf{x} − \hat{\mathbf{r}}||_2^2 − \beta \mathcal{D}_{KL}(q_\theta(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \qquad (5)$$

The distribution $p(\mathbf{z})$ is chosen according to the principle of maximum entropy: since the latent variables are in the range $(-\infty, +\infty)$, the Gaussian distribution is the most appropriate for this case. Each latent variable is then constrained to have a Gaussian distribution with zero mean and one variance. Similar to VAE, we cannot train the encoder by directly sampling the mean and variance of the latent vector. Instead, we employ the re-parametrization trick: we sample a random vector ϵ from a normal distribution, the latent vector is represented as $\mathbf{z_x} = \mu_x + \epsilon \odot \sigma_x^2$, with $\odot$ the element-wise product. Because the role of $\mathbf{z_r}$ is only to reconstruction, it should be noted that we do not have a random normal vector for this variable. Figure 3 summarizes training a VAE with feature alignment.

3.4. Improving the quality of the features (VFA-GAN)

As will be shown in the results section, the images extracted using the feature alignment trained with VAE are blurry, due to the variational autoencoder nature (Rezende and Viola, 2018). We add a generator network G and a discriminator network D to the images generated by the technique to improve their quality. In this manner, the optimized random vector functions as a second latent vector, thereby sampling a more complex distribution from the latent representation. This generator network is similar to the refiner network described in Atapattu and Rekabdar (2019), which takes an image as input and outputs an improved version of it.

The optimized random input is fed into the generator, which then generates a new output that is compared to the input $\mathbf{x}$. The discriminator is trained as a generative
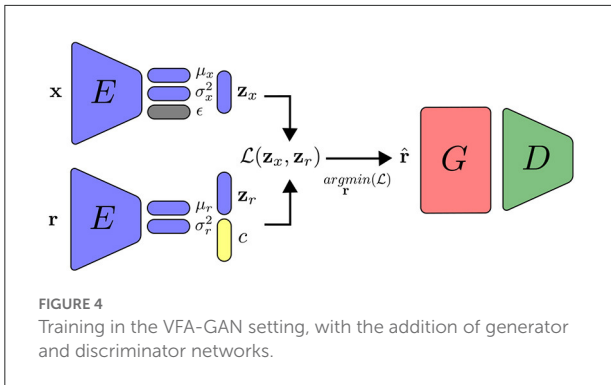
FIGURE 2
Approximation of reversibility. **(A)** The network trained without feature alignment can only recover partially the inputs. **(B)** The network trained with feature alignment can retrieve all the inputs.

FIGURE 3
Variational autoencoder with feature alignment. **Left**: training the encoder to reconstruct the inputs x. **Right**: we randomly sample a normal vector ϵ to generate new data statiscally simillar to the training data. Note that the main different between encoders with FA and VFA is on the latent representation.

adversarial network, that assesses the likelihood that $G(\hat{\mathbf{r}})$ is genuine or fake (that is, i.e., whether it comes from training data or not). The generator is updated by receiving gradients from the discriminator. For more stable training, we use the least square loss for the discriminator (Mao et al., 2017). Alternatively, it may be possible to use the Wasserstein GAN formulation (Arjovsky et al., 2017), which replaces the discriminator with a critic network that measures the score of the "*realness*" of an image.

We propose using a random schedule for the variable β in order to reduce the potential effects that could be caused by the posterior collapse problem in VAEs (Lucas et al., 2019; Havrylov and Titov, 2020; Takida et al., 2021) and to maintain a balance

with the reconstruction loss. We take a sample from a uniform distribution $\beta \leftarrow \mathcal{U}(0,1)$ for every example that we go through in the training process.

Although pixel-level loss is typically used to optimize image reconstruction, high-level data properties can also be considered. Perceptual loss (Johnson et al., 2016) is a type of measurement that compares the output of the reconstruction with the original image at high-level neurons (presented near the end of the network). The mean and variance layers from the encoder network are used in this case as the perceptual loss, requiring the reconstruction to have the same statistical properties as the original input.

FIGURE 4
Training in the VFA-GAN setting, with the addition of generator and discriminator networks.

The final losses, for the encoder, generator, and discriminator are shown in Equations (6)–(8), respectively:

$$\mathcal{L}_E = ||\mathbf{x} - \hat{\mathbf{r}}||_2^2 + \beta \mathcal{D}_{KL}(q_\theta(\mathbf{z}|\mathbf{x})||p(\mathbf{z})), \tag{6}$$

$$\mathcal{L}_G = ||1 - D(G(\hat{\mathbf{r}}))||_2^2 + \lambda(||\mu(\mathbf{x}) - \mu(G(\hat{\mathbf{r}}))||_2^2$$
$$+ ||\sigma^2(\mathbf{x}) - \sigma^2(G(\hat{\mathbf{r}}))||_2^2), \tag{7}$$

$$\mathcal{L}_D = ||1 - D(\mathbf{x})||_2^2 + || - 1 - D(G(\hat{\mathbf{r}}))||_2^2. \tag{8}$$

with λ a hyperparameter that weights the perceptual loss contribution.

If training data contains additional information, such as labels, it is possible to train a neural network simultaneously with supervised training for specific related tasks such as conditional generation, in which samples must correspond to a desired class. We can condition the latent variables to have different distributions from different classes. The output of the network is trained using the Gaussian distribution $\mathbf{z} \sim Q(z_i, c_i)$, with c_i being a one-hot vector containing the class information. Similar to the work done in Ardizzone et al. (2019), we couple a linear classification layer on top of the network, parallel to the mean and variance layers. Since the class layer is linear, we can choose a higher value for the one-hot vector during inference time to emphasize the selected class. The improvements made on the variational autoencoders with feature alignment discussed in this section are illustrated in Figure 4.

3.5. Local feature alignment

The rules for feature alignment were presented as a global rule: the auxiliary loss is defined with the output layer (with the pair z_x and z_r) and the loss is defined with the input layer (with the pair x and $\hat{r}$), allowing full exchange of information between all layers. However, we can reformulate this rule with local losses, similar to target propagation rules (Bengio, 2014; Farias and Maziero, 2018; Ororbia et al., 2018): the auxiliary loss and loss are defined as the interaction between two connected layers only (or even individual neurons), as follows: for each

layer l, from the first to the last, we activate x_{l+1} from its inputs x_l and store a second activation from a random input r_l with the same dimension. We then optimize the random input r_l with an auxiliary loss between activation of the random output r_{l+1} and the true output x_{l+1}. Finally, the parameters of the chosen layer θ_l are updated by optimizing the loss between the reconstruction $\hat{r}_l$ and true input x_l. This technique of local training is summarized in Algorithm 2 and illustrated in Figure 5.
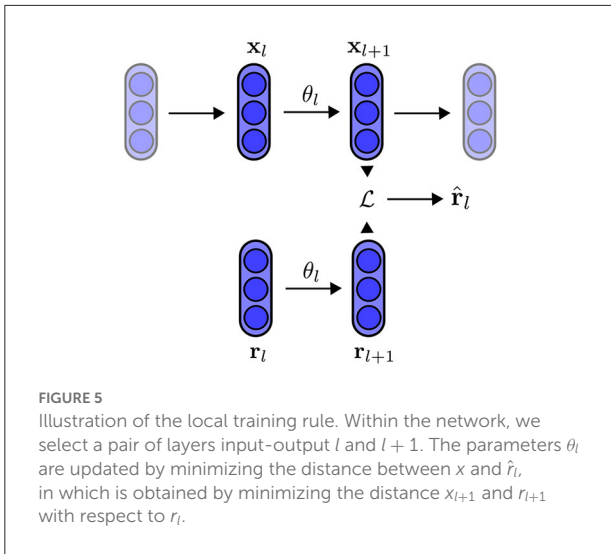
```
1: for l=0, L do                    ▷ for each layer
2:      z_x^l = E(x; θl)
3:      initialize r = x.shape
4:      t = 0
5:      while t < T do
6:          z_r^l = E(r; θl)
7:          ℒl = ||z_x^l − z_r^l||_2^2
8:          r = r − τ ∂ℒl/∂r
9:          t = t + 1
10:     end while
11:     r̂ = r
12:     Cl = ||x − r̂||^2
13:     update θl by optimizing Cl
14:     x = z_x^l.detach
15: end for
```

Algorithm 2. Training with local feature alignment.

Each layer of the neural network trains its parameters to become a predictive machine by attempting to predict the inputs using knowledge of the outputs. The local learning constraint has a greater impact on the non-linearity of a neural network trained in this manner. Local rules can only rely on very strict information content available, whereas backpropagation can adjust all network parameters so that the feature reconstructs the input. Non-reversible functions, like the ReLU function, propagate loss of information, resulting in low-fidelity reconstructions. In order to retain as much information as possible, a non-linear function must be carefully chosen. The function *inverse hyperbolic sine* ($arc \sinh(x) = \ln(x + \sqrt{1 + x^2})$)), is similar to the *hyperbolic tangent* near zero and logarithmic at large (absolute) values. This function has the properties of being fully invertible, zero-centered mean, unbounded, continuously differentiable and its gradient does not vanish as fast as for tanh. These properties make *arc* sinh a good candidate function for local training.

Similar to how non-local feature alignment is typically done, we must propagate the information backward from the output to the input layer by layer at inference time. However, the non-linear function will be crucial in this situation because it must be reversible in order to approximate reversibility. This is accomplished by applying the inverse of the non-linear function

FIGURE 5
Illustration of the local training rule. Within the network, we select a pair of layers input-output l and $l+1$. The parameters θ_l are updated by minimizing the distance between x and $\hat{r}_l$, in which is obtained by minimizing the distance x_{l+1} and r_{l+1} with respect to r_l.

after each layer that makes use of the function, following the input of the latent vector. The Algorithm 3 provides a summary of this procedure.

```
1:  sample z_x^L
2:  for l=L, 0 do
3:      initialize r = x.shape with x as z_x^l = E(x; θ_l)
4:      t = 0
5:      while t < T do
6:          z_r^l = E(r; θ_l)
7:          L_l = ||z_x^l − z_r^l||_2^2
8:          r = r − τ ∂L_l/∂r
9:          t = t + 1
10:     end while
11:     r̂ = r
12:     if layer = non-linear function f then
13:         r̂ = f^{-1}(r̂)
14:     end if
15:     z_x^l = r̂
16: end for
```

Algorithm 3. Reconstruction with local feature alignment.

4. Implementation details

The encoder network consists of a series of convolutional layers, similar to the AlexNet architecture (Krizhevsky et al., 2012), but with stride one and two for down-scaling, instead of maxpool, with LeakyReLU activation. The generator network has three convolutional layers. The discriminator network has the same architecture as the encoder, except for the last layer that outputs a single value. Only the generator utilizes batch normalization after each convolution. All convolutions have

kernel size $k = 3$. Details of the networks can be found in Tables 7–12 in Appendix B for MNIST, CIFAR-10, CelebA and STL-10, respectively.

We use the Adam optimizer (Kingma and Ba, 2017) with learning rate $\eta = 0.00001$ and batch size 128. The parameters of the encoder and generator networks are initialized with orthogonal initialization (Saxe et al., 2014; Hu et al., 2020). We set the hyperparameter $\lambda = 0.01$ and sample β from a uniform distribution, with a different random value for each training example. We set the hyper-parameter $\tau = 1$ for the reconstruction of the input at one-shot $T = 1$.

From Appendix B, we have that the loss becomes unstable when the weights $w^2 > 2$, so we restrict the weights to the range $-\sqrt{2} \leq w \leq \sqrt{2}$ by clamping then, as shown in Equation (9).

$$w = \begin{cases} -\sqrt{2} & \text{if } w < -\sqrt{2}, \\ \sqrt{2} & \text{if } w > \sqrt{2}, \\ w & \text{otherwise.} \end{cases} \tag{9}$$

We also report the results of the modified feature alignment for local training as a proof of concept by training an encoder for reconstruction from a latent vector. For GAN, when used for reconstruction, we search the latent space that leads to most similar images by optimizing $\text{argmin}_z ||x - G(z)||_2^2$.

5. Results

We compare the results against traditional variational autoencoders and generative adversarial networks. The results show the reconstruction of the inputs using feature alignment and generator applied to the generated input. Additionally, we also show random samples from the generator network. Furthermore, we display the model size and inference time for each dataset in the next section. All models were trained and evaluated on an Nvidia RTX 2070 graphics processing unit (GPU) card.

We measured the quality of the results by using the Fréchet Inception Distance (FID) (Heusel et al., 2018; Seitzer, 2020). The FID score is calculated by extracting the activation of the global spatial pooling layer of a pre-trained Inception V3 model (Szegedy et al., 2016), for equally numbered images from the dataset (here we choose 10,000 images) and sampled from a generator model, as shown in Equation (10).

$$FID = ||\mu_1 - \mu_2||_2^2 + tr(\Sigma_1 + \Sigma_2 - 2\sqrt{\Sigma_1 \Sigma_2}) \tag{10}$$

with μ the mean of activations, Σ the covariance matrix and tr the trace function. The FID score, as opposed to pixel-level comparisons, compares the similarity of images at a high level in the feature layers, where significant patterns can be identified. This is in contrast to comparisons that are made at the pixel level. Due to the fact that FID is measured on a collection of

TABLE 2 FID scores across four image datasets.

Method	MNIST	CelebA	CIFAR-10	STL-10
VAE	39.84 ± 0.15	84.84 ± 0.10	163.59 ± 0.37	201.59 ± 0.46
GAN	21.50 ±2.79	32.85 ± 1.22	63.39 ± 0.62	247.35 ± 2.43
VFA (ours)	120.02 ± 1.11	143.51 ± 1.04	209.32 ± 1.96	259.96 ± 2.46
VFA − GAN (ours)	41.24 ± 2.71	132.18 ± 2.73	73.20 ± 2.67	167.26 ± 5.18

Mean and standard deviation from three trials. Lower is better. We can see that the generator improves the quality of the samples by lowering the FID score from sampling variational autoencoders trained with feature alignment.
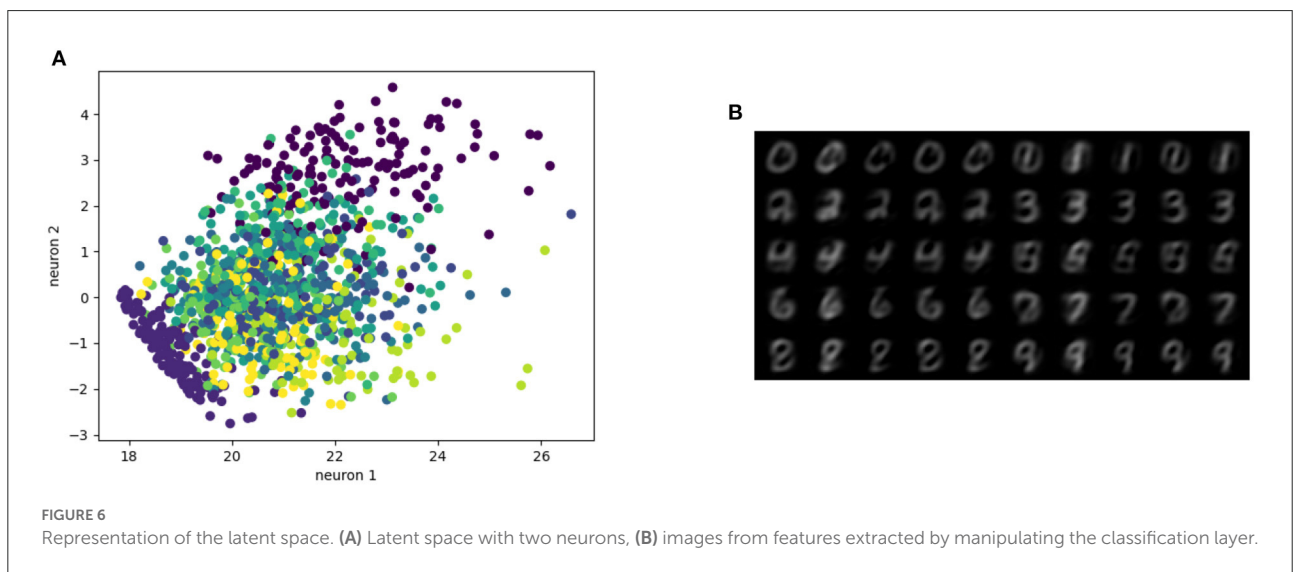

FIGURE 6
Representation of the latent space. **(A)** Latent space with two neurons, **(B)** images from features extracted by manipulating the classification layer.

images, we are able to make a comparison between the statistics of the distributions found in natural images (or any other set of images that may be desired) and those produced by a generative method. Because it operates in a manner analogous to that of a distance metric, values that are lower indicate that the generated images are statistically more comparable to either the training or test data.

Table 2 shows the average FID score results for three different initializations. We can see that across the four datasets, feature alignment has higher scores, which indicates that, in comparison to the other approaches, it has a lower sampling quality. When a generative network is used, however, feature alignment can achieve scores that are more comparable to those achieved by GANs.

The results of each dataset section below demonstrate that VFA has a lower sample quality and, thus, a greater FID. This behavior is primarily caused by two factors: first, VFA has the same limitations as variational auto-encoders, in which sampling is constrained by the latent layer and tends to produce blurry images to some extent as a result of the loss attempting to approximate the distribution on the latent space to a normal distribution. Second, in addition to optimizing reconstruction and approaching the normal distribution, the loss of VFA must
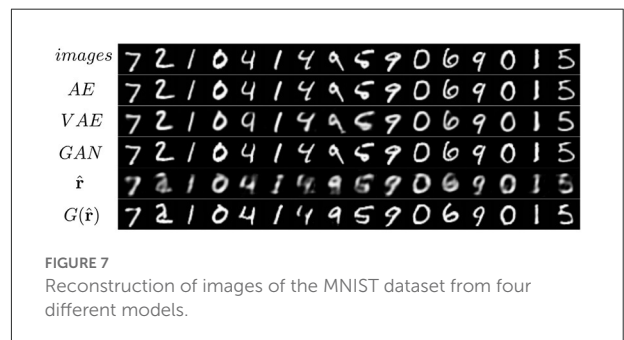

FIGURE 7
Reconstruction of images of the MNIST dataset from four different models.

also optimize reversibility when the encoders of a VAE and VFA are set to have the same size. The GAN formulation and publications concerning the FID metric both demonstrate that both VFA and VAE contain losses that act on pixel space, which is not always the optimal measure to produce sharp images. Since the generator is trained with a discriminator and both have losses that function on a different space than the pixels, connecting the generator network to enhance the quality of the reconstructed inputs leads to better samples and a lower FID.

FIGURE 8
Left: random samples. **Right**: interpolation among four images reconstructed from the dataset.

TABLE 3 Size and execution time required to draw a single sample using multiple models trained on the MNIST dataset.

Model	Size ($\times 10^5$ parameters)	Time ($\times 10^{-4}$ s)
VAE	27.40	9.44
GAN	13.44	5.73
VFA	14.0	11.66
VFA-GAN	14.16	20.59

VFA takes approximately the same amount of time as VAE, but only half the number of parameters.

5.1. MNIST

The MNIST dataset (Lecun et al., 1998) is a collection of 60,000 grayscale images with size 28×28 pixels that contain hand drawings of digits from zero to nine. Figure 6A shows the latent space trained on two neurons as outputs, we can see that the network attempts to cluster the images to similarity, while Figure 6B shows the reconstruction for a fixed latent vector but varying a trained classified output vector according to the labels of the dataset. Figure 7 shows the reconstruction of images by the features and with the generator applied to them, compared with traditional AE, VAE, and GAN. Even though the reconstructions are frequently noisy, the generator can sharpen the images to make them more similar to the original inputs. Figure 8 shows random samples from the generator with a class layer coupled to the encoder. To show the consistency of transitions on the latent space, the same image also features the interpolation of the latent vector between four pairs of images.

The size of the model and the amount of time required to draw a single sample are provided in the Table 3. The size of a feature algorithm is approximately half that of a variational autoencoder because it trains a network without a decoder, but its execution time is roughly the same due to the backward pass used to update the random input.

5.2. CelebA

The CelebA dataset (Liu et al., 2015) is a collection of 202,600 images of celebrity faces. The images were resized to 64×64 pixels. Figures 9, 10 show the reconstruction and sampling with interpolation between samples, respectively. Without the perceptual loss (with $\lambda = 1$), we noticed a failure on the convergence of the generator network, resulting in samples containing only noise.

The size and duration of sampling a single image for various models are displayed in Table 4. As before, we can see that VFA takes about the same amount of time despite being half the size.

5.3. CIFAR-10

The CIFAR-10 dataset (Krizhevsky and Hinton, 2009) contains 70,000 natural images with size 32×32 pixels across 10 different classes. Figure 11 shows the reconstruction of images from the dataset. While the features do approximate the original inputs, the transformation of the generator tends to be more dissimilar due to its loss being dependent only on the adversarial contribution ($\lambda = 0$). Just as before, Figure 12 shows random samples and interpolation, which show a diversity of images, albeit less perceptual similar to the original dataset.

Table 5 shows the size and time of sampling for different models. Since VFA is very similar to VAE, but without the decoder, we can see that it maintains the half-size pattern for the same amount of time, due to the backward pass.

5.4. STL-10

The STL-10 dataset (Adam et al., 2011) is a subset of the ImageNet dataset that contains 100,000 unlabeled images, and a additional of 500 labeled images for training and 800 images

FIGURE 9
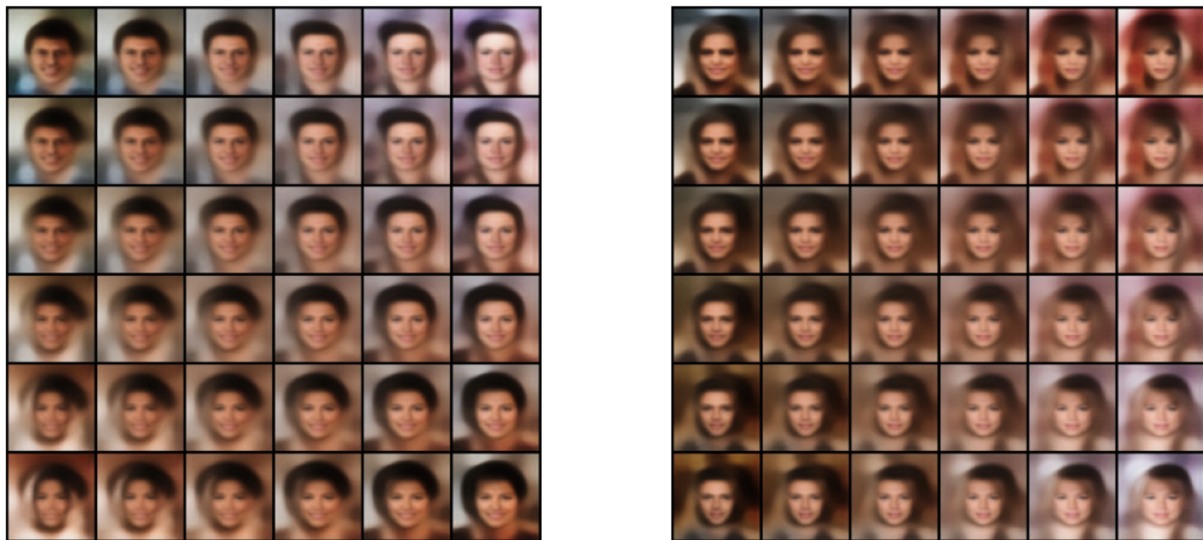Reconstruction of images of the CelebA dataset from four different models.

FIGURE 10
Two sets of interpolation among four random sampled images.

TABLE 4 Size and execution time required to draw a single sample using multiple models trained on the CelebA dataset.

Model	Size ($\times 10^5$ parameters)	Time ($\times 10^{-4}$ s)
VAE	37.48	19.41
GAN	18.48	12.45
VFA	19.04	22.79
VFA-GAN	19.24	41.71

VFA takes approximately the same amount of time as VAE, but only half the number of parameters.

for testing. This dataset is mostly used for unsupervised tasks, but since in this work we are interested in image generation, we used only the set of unlabeled images, resized to 64×64 pixels.

The reconstruction results are shown in Figure 13. We can see that the reconstruction $\hat{r}$ from VFA is visually similar to VAE, but has slightly better fidelity to shapes.

Random samples from VFA-GAN are shown in Figure 14. We also show, in Table 6, the size of each model and the time required to draw samples. Note that the size of the model *VFA-GAN* includes both the encoder and generator networks.

It is important to note that feature alignment is not expected to outperform the reconstruction and sample qualities of VAEs and GANs. Because the reversibility condition is a constraint on neural network optimization, which must thus balance the reversibility cost with other losses. Nevertheless, we compare the reconstruction L_2 loss with other networks to analyze how different each network is compared to the same metric. These results are shown in Figure 15.
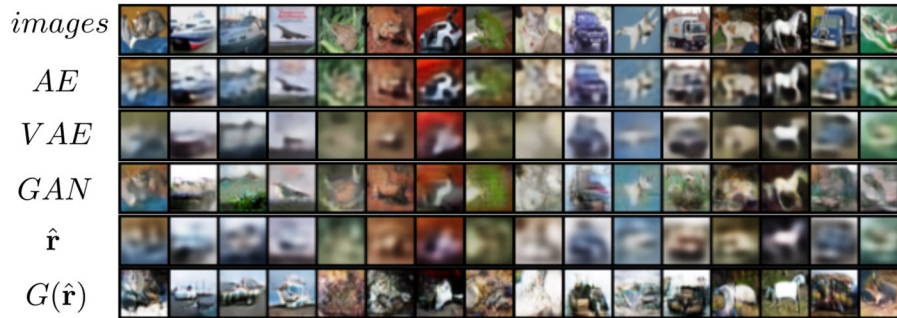
FIGURE 11
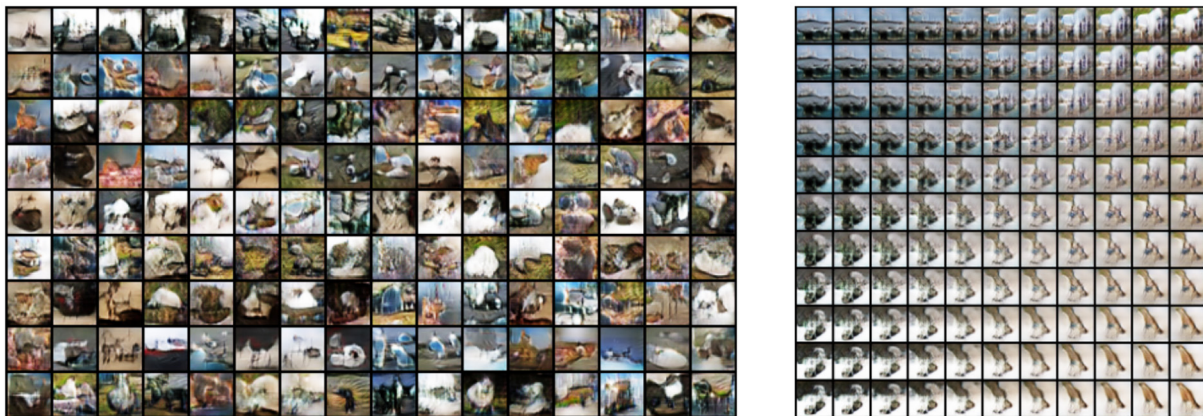Reconstruction of images on CIFAR-10.

FIGURE 12
Left: random samples. **Right**: interpolation among four images reconstructed from the dataset.

TABLE 5 Size and execution time required to draw a single sample using multiple models trained on the CIFAR-10 dataset.

Model	Size ($\times 10^5$ parameters)	Time ($\times 10^{-4}$ s)
VAE	97.55	10.01
GAN	47.48	7.20
VFA	50.28	11.25
VFA-GAN	51.95	21.56

VFA takes approximately the same amount of time as VAE, but only half the number of parameters.

FIGURE 13
Reconstruction of images on STL-10 from different models.

Given that it is optimized directly for reconstruction, autoencoders have the lowest loss, which is to be expected given the nature of the network. Because they are optimized with the same amount of loss, the reconstructions from feature alignment should be compared to VAEs and the generator should be compared to GAN. It is clear from this that the lack of perceptual loss on the gene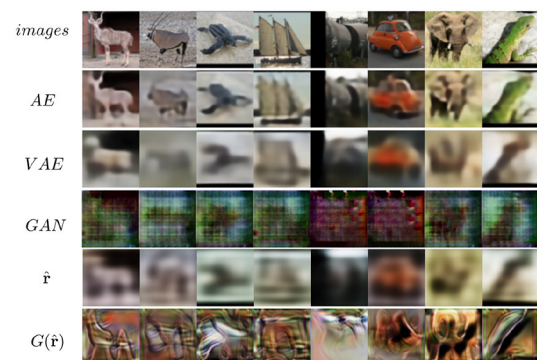rator network (for the CIFAR-10 dataset) has a negative impact on the reconstruction ability (without first optimizing the latent vector).
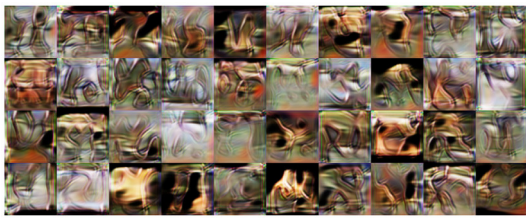
FIGURE 14
Random samples from VFA-GAN of images on STL-10.

TABLE 6 Size and execution time required to draw a single sample using multiple models trained on the STL-10 dataset.

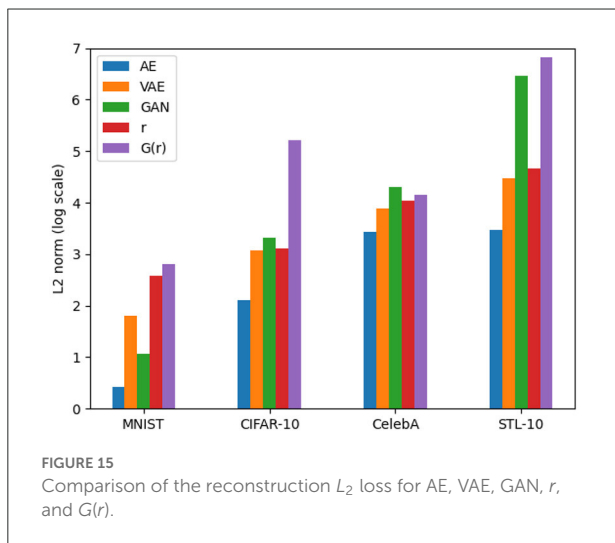| Model	Size ($\times 10^5$ parameters)	Time ($\times 10^{-2}$ s)
VAE	254.22	1.51
GAN	32.74	1.50
VFA	221.48	1.72
VFA-GAN	229.90	2.18

FIGURE 15
Comparison of the reconstruction L_2 loss for AE, VAE, GAN, r, and $G(r)$.

5.5. Local feature alignment

In this section, we present the results obtained by an encoder that was trained for reconstruction using the local feature alignment training. Figures 16A–D show reconstruction pairs for the MNIST, CIFAR-10, CelebA and STL-10 datasets, respectively. We can observe that local training can reconstruct images even though the layers do not receive any information from the reconstruction loss of images. This can be attributed to the same reason as non-local feature alignment: the weights form an orthogonal matrix that attempts to reverse information between layers as much as possible, that is only limited by

the network capacity, which is directly related to the number of neurons.

6. Conclusions

We presented feature alignment, a technique to approximate reversibility in neural networks. By optimizing the features to match the inputs, we trained an encoder to predict its input, given an output. For a simple case, we showed that it is possible to recover the inputs given only the outputs by adding latent variables, which are optimized only with the reversibility loss. We can generate new samples with the same statistical distribution as the training data by coupling a probabilistic layer with the same formulation as the variational autoencoders. We combined the generative adversarial network method by coupling a generator and a discriminator network to the images generated by the method to improve the quality of the generated samples, which suffer from noise effects. We also demonstrated that the technique can be modified to use a local training rule instead of backpropagation, which has the advantage of using less memory for training and extracting gradients from neural networks.

Mathematical analysis on the convergence of the proposed technique shows that the weights converge to a pseudo-inverse matrix, which justifies the convergence of a network trained in this way to map its outputs back to its inputs. Since the bottlenecks do not permit a one-to-one relationship, the restriction is the architecture of the network itself.

We used the technique to reconstruct and generate images from the datasets MNIST, CIFAR-10, CelebA and STL-10. The results demonstrate that the features can approximate the inputs. Despite the fact that it cannot improve on the sampling quality of other current generative techniques, reversibility can be advantageous when a mapping of the outputs back to their inputs is desired. The CIFAR-10 and STL-10 datasets are notoriously difficult due to the small image size and high variance, resulting in samples with a high FID measure.

The primary shortcoming of the approach is that it places a restriction of reversibility on the parameters of a neural network. This forces the parameters to be balanced between reconstruction or sampling and reversibility, which in turn reduces the quality of the images that are produced. When an encoder is trained to compress an image into a latent representation, which must approximate a normal distribution, and also reconstruct the input given this representation, this can be seen as an additional loss to optimize. An encoder is trained to accomplish these two tasks simultaneously. Therefore, in order to recover the inputs, the trainable parameters of a network not only need to minimize some loss with respect to the forward propagation of information, but they also need to reduce some loss with respect to the backward propagation of information. As a direct consequence of this, there is a limited number of
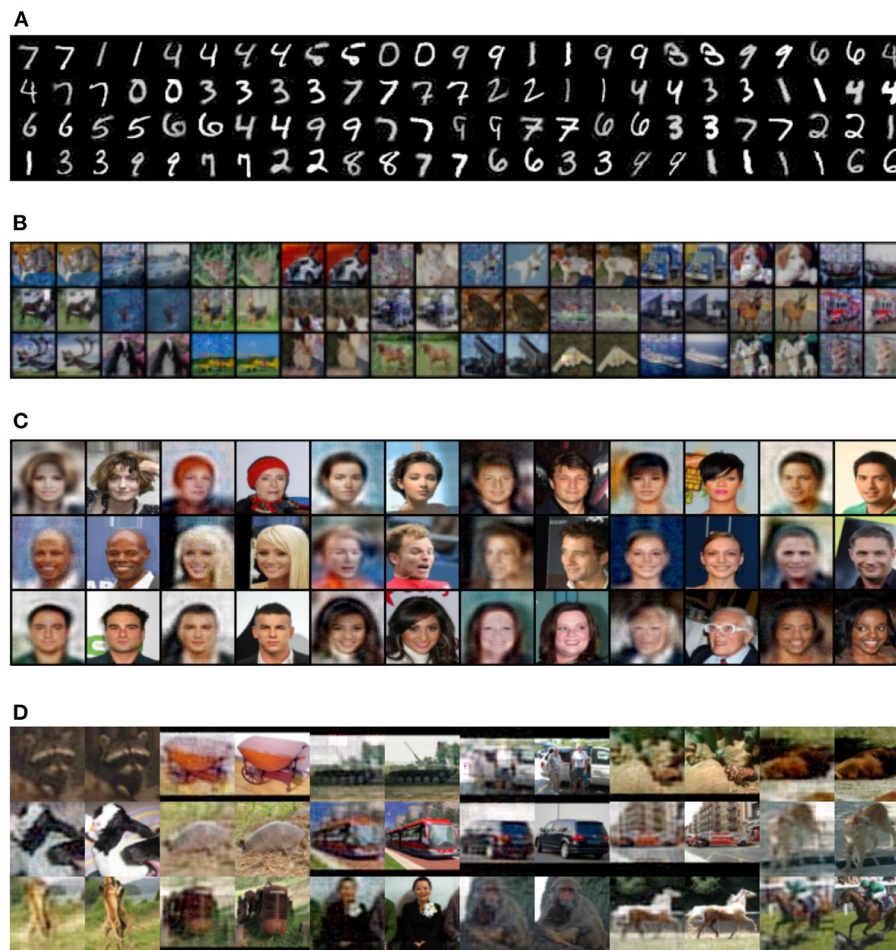
FIGURE 16
Local feature alignment. Each pair of images contains the reconstruction and original, respectively. **(A)** MNIST, **(B)** CIFAR-10, **(C)** CelebA, and **(D)** STL-10.

practicable optimal configurations that the trainable parameters are capable of reaching.

We can use label information for conditional sample generation by connecting a classification layer to the encoder network. Furthermore, the results of local training suggest that we can train neural networks without using global loss function feedback, which is an important area of application of this technique.

The proposed method can be interpreted in a way that places it somewhere in the middle of VAEs and GANs. The complete architecture, which is comprised of an encoder and a generator network, possesses more complex latent vectors that may be exploited and generates samples that are crisper than those produced by VAEs. Feature alignment is a technique that can be implemented across a broad variety of neural network designs so long as the architecture of the neural networks being utilized can be entirely differentiated end to end.

As a result, it offers the possibility of making modifications, which could potentially lead to an improvement in the results in general. For instance, utilizing residual networks, such as ResNets (He et al., 2016) for the encoder and U-Net (Ronneberger et al., 2015) for the generator network, since the latter can transfer the dimension of the input onto itself, are two examples of how this may be done to both improve the flow of information and reduce the influence of disappear gradients.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found at: The links from the datasets are listed on the code repository: https://github.com/tiago939/feature_alignment.

Author contributions

All authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

Funding

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

The reviewer JV declared a shared affiliation, with no collaboration with the authors at the time of the review.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frai.2022.1025148/full#supplementary-material

References

Adam, C., Andrew, N., and Honglak, L. (2011). "An analysis of single layer networks in unsupervised feature learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Vol 15*, (PMLR), 215–223.

Ardizzone, L., Kruse, J., Wirkert, S., Rahner, D., Pellegrini, E. W., Klessen, R. S., et al. (2019). Analyzing inverse problems with invertible neural networks. *arXiv:1808.04730 [cs, stat]*. doi: 10.48550/arXiv.1808.04730

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. *arXiv:1701.07875 [cs, stat]*.

Atapattu, C., and Rekabdar, B. (2019). "Improving the realism of synthetic images through a combination of adversarial and perceptual losses," in *2019 International Joint Conference on Neural Networks (IJCNN)* (Budapeste), 1–7.

Baird, L., Smalenberger, D., and Ingkiriwang, S. (2005). "One-step neural network inversion with PDF learning and emulation," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005, Vol. 2* (Montreal, QC: IEEE), 966–971.

Baldi, P., and Sadowski, P. (2016). A theory of local learning, the learning channel, and the optimality of backpropagation. *Neural Netw.* 83, 51–74. doi: 10.1016/j.neunet.2016.07.006

Behrmann, J., Grathwohl, W., Chen, R. T. Q., Duvenaud, D., and Jacobsen, J.-H. (2019). Invertible residual networks. *arXiv:1811.00995 [cs, stat]*. doi: 10.48550/arXiv.1811.00995

Bengio, Y. (2014). How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv:1407.7906 [cs]*. doi: 10.48550/arXiv.1407.7906

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., et al. (2018). Understanding disentangling in β-VAE. *arXiv:1804.03599 [cs, stat]*. doi: 10.48550/arXiv.1804.03599

Chen, T., Xu, B., Zhang, C., and Guestrin, C. (2016). Training deep nets with sublinear memory cost. *arXiv:1604.06174*. doi: 10.48550/arXiv.1604.06174

Dauvergne, B., and Hascoët, L. (2006). "The data-flow equations of checkpointing in reverse automatic differentiation," in *Computational Science 96 ICCS 2006. ICCS 2006. Lecture Notes in Computer Science, vol 3994*, eds V. N. Alexandrov, G. D. van Albada, P. M. A. Sloot, and J. Dongarra (Berlin; Heidelberg: Springer). doi: 10.1007/11758549_78

Doersch, C. (2021). Tutorial on variational autoencoders. *arXiv:1606.05908 [cs, stat]*. doi: 10.48550/arXiv.1606.05908

Donahue, J., Krähenbühl, P., and Darrell, T. (2017). Adversarial feature learning. *arXiv:1605.09782 [cs, stat]*. doi: 10.48550/arXiv.1605.09782

Dong, X., Yin, H., Alvarez, J. M., Kautz, J., and Molchanov, P. (2021). Deep neural networks are surprisingly reversible: a baseline for zero-shot inversion. *Techn. Rep.* arXiv:2107.06304. doi: 10.48550/arXiv.2107.06304

Dosovitskiy, A., and Brox, T. (2016). Generating images with perceptual similarity metrics based on deep networks. *arXiv:1602.02644 [cs]*. doi: 10.48550/arXiv.1602.02644

Ellis, C. A., Sendi, M. S., Miller, R., and Calhoun, V. (2021). "A novel activation maximization-based approach for insight into electrophysiology classifiers," in *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (Houston, TX: IEEE), 3358–3365.

Fan, F., Xiong, J., Li, M., and Wang, G. (2021). On interpretability of artificial neural networks: a survey. *arXiv:2001.02522 [cs, stat]*. doi: 10.1109/TRPMS.2021.3066428

Farias, T. S., and Maziero, J. (2018). Gradient target propagation. *arXiv:1810.09284 [cs]*. doi: 10.48550/arXiv.1810.09284

Gao, F., and Zhong, H. (2020). Study on the large batch size training of neural networks based on the second order gradient. *arXiv:2012.08795 [cs]*. doi: 10.48550/arXiv.2012.08795

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., and Kagal, L. (2019). Explaining explanations: an overview of interpretability of machine learning. *arXiv:1806.00069 [cs, stat]*. doi: 10.1109/DSAA.2018.00018

Gomez, A. N., Ren, M., Urtasun, R., and Grosse, R. B. (2017). The reversible residual network: backpropagation without storing activations. *arXiv:1707.04585 [cs]*. doi: 10.48550/arXiv.1707.04585

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial networks. *arXiv:1406.2661 [cs, stat]*. doi: 10.48550/arXiv.1406.2661

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2018). FFJORD: free-form continuous dynamics for scalable reversible generative models. *arXiv:1810.01367 [cs, stat]*. doi: 10.48550/arXiv.1810.01367

Gui, J., Sun, Z., Wen, Y., Tao, D., and Ye, J. (2020). A review on generative adversarial networks: algorithms, theory, and applications. *arXiv:2001.06937 [cs, stat]*. doi: 10.48550/arXiv.2001.06937

Guo, W., Fouda, M. E., Eltawil, A. M., and Salama, K. N. (2022). BackLink: supervised local training with backward links. *Techn. Rep.* arXiv:2205.07141. doi: 10.48550/arXiv.2205.07141

Havrylov, S., and Titov, I. (2020). Preventing posterior collapse with levenshtein variational autoencoder. *arXiv:2004.14758 [cs, stat].* doi: 10.48550/arXiv.2004.14758

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV: IEEE), 770–778.

Hebb, D. O. (1949). The organization of behavior. *Wiley Brain Res. Bull.* 50, 437. doi: 10.1016/S0361-9230(99)00182-3

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2018). GANs trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv:1706.08500 [cs, stat].* doi: 10.48550/arXiv.1706.08500

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., et al. (2016). "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework," in *International Conference on Learning Representations.*

Ho, J., Jain, A., and Abbeel, P. (2020). "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems, Vol. 33*, eds H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Vancouver, CA: Curran Associates, Inc.), 6840–6851.

Hu, W., Xiao, L., and Pennington, J. (2020). Provable Benefit of orthogonal initialization in optimizing deep linear networks. *arXiv:2001.05992 [cs, math, stat].* doi: 10.48550/arXiv.2001.05992

Ismail, A. A., Bravo, H. C., and Feizi, S. (2021). "Improving deep learning interpretability by saliency guided training," in *Advances in Neural Information Processing Systems*, eds A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan (Virtual Conference).

Isomura, T., and Toyoizumi, T. (2016). A local learning rule for independent component analysis. *Sci. Rep.* 6, 28073. doi: 10.1038/srep28073

Isomura, T., and Toyoizumi, T. (2018). Error-gated hebbian rule: a local learning rule for principal and independent component analysis. *Sci. Rep.* 8, 1835. doi: 10.1038/s41598-018-20082-0

Jing, K., and Xu, J. (2019). A survey on neural network language models. *arXiv:1906.03591 [cs].* doi: 10.48550/arXiv.1906.03591

Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. *arXiv:1603.08155 [cs].* doi: 10.1007/978-3-319-46475-6_43

Kingma, D. P., and Ba, J. (2017). Adam: a method for stochastic optimization. *arXiv:1412.6980 [cs].* doi: 10.48550/arXiv.1412.6980

Kingma, D. P., and Dhariwal, P. (2018). Glow: generative flow with invertible 1x1 convolutions. *arXiv:1807.03039 [cs, stat].* doi: 10.48550/arXiv.1807.03039

Kingma, D. P., and Welling, M. (2014). Auto-encoding variational bayes. *arXiv:1312.6114 [cs, stat].* doi: 10.48550/arXiv.1312.6114

Kingma, D. P., and Welling, M. (2019). An introduction to variational autoencoders. *arXiv:1906.02691 [cs, stat].* doi: 10.1561/9781680836233

Kobyzev, I., Prince, S. J. D., and Brubaker, M. A. (2020). Normalizing flows: an introduction and review of current methods. *arXiv:1908.09257 [cs, stat].* doi: 10.48550/arXiv.1908.09257

Krizhevsky, A., and Hinton, G. (2009). *Learning Multiple Layers of Features From Tiny Images* (Master's thesis). Department of Computer Science, University of Toronto.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems, Vol. 25* (Nevada).

Krotov, D., and Hopfield, J. J. (2019). Unsupervised learning by competing hidden units. *Proc. Natl. Acad. Sci. U.S.A.*116, 7723–7731. doi: 10.1073/pnas.1820458116

Kumar, R., Purohit, M., Svitkina, Z., Vee, E., and Wang, J. (2019). "Efficient rematerialization for deep networks," in *Advances in Neural Information Processing Systems, Vol. 32*, eds H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Vancouver, CA: Curran Associates, Inc.).

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric. *arXiv:1512.09300 [cs, stat].* doi: 10.48550/arXiv.1512.09300

Laskin, M., Metz, L., Nabarro, S., Saroufim, M., Noune, B., Luschi, C., et al. (2021). Parallel training of deep networks with local updates. *arXiv:2012.03837.* doi: 10.48550/arXiv.2012.03837

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324. doi: 10.1109/5.726791

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2011). Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Commun. ACM.* 54, 95–103. doi: 10.1145/2001269.2001295

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., and Hinton, G. (2020). Backpropagation and the brain. *Nat. Rev. Neurosci.* 21, 335–346. doi: 10.1038/s41583-020-0277-3

Lindsey, J., and Litwin-Kumar, A. (2020). Learning to learn with feedback and local plasticity. *arXiv:2006.09549 [cs, q-bio].* doi: 10.48550/arXiv.2006.09549

Linnainmaa, S. (1976). Taylor expansion of the accumulated rounding error. *BIT Num. Math.* 16, 146–160. doi: 10.1007/BF01931367

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)* (Santiago).

Lucas, J., Tucker, G., Grosse, R., and Norouzi, M. (2019). Don't blame the ELBO! A linear VAE perspective on posterior collapse. *arXiv:1911.02469 [cs, stat].* doi: 10.48550/arXiv.1911.02469

Mahendran, A., and Vedaldi, A. (2016). Visualizing deep convolutional neural networks using natural pre-images. *Int. J. Comput. Vis.* 120, 233–255. doi: 10.1007/s11263-016-0911-8

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., and Smolley, S. P. (2017). "Least squares generative adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)* (Venice: IEEE), 2813–2821.

Millidge, B., Tschantz, A., Seth, A. K., and Buckley, C. L. (2020). Activation relaxation: a local dynamical approximation to backpropagation in the brain. *arXiv:2009.05359 [cs, q-bio].* doi: 10.48550/arXiv.2009.05359

Nair, V., and Hinton, G. E. (2010). "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, 807–814.

Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., and Clune, J. (2016a). Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *arXiv:1605.09304 [cs].* doi: 10.48550/arXiv.1605.09304

Nguyen, A., Yosinski, J., and Clune, J. (2016b). Multifaceted feature visualization: uncovering the different types of features learned by each neuron in deep neural networks. *arXiv:1602.03616 [cs].* doi: 10.48550/arXiv.1602.03616

Olah, C., Mordvintsev, A., and Schubert, L. (2017). Feature visualization. *Distill* 2, e7. doi: 10.23915/distill.00007

Ororbia, A. G., Mali, A., Kifer, D., and Giles, C. L. (2018). Conducting credit assignment by aligning local representations. *arXiv:1803.01834 [cs, stat].* doi: 10.48550/arXiv.1803.01834

Papamakarios, G. (2019). Neural density estimation and likelihood-free inference. *arXiv:1910.13233 [cs, stat].* doi: 10.48550/arXiv.1910.13233

Rezende, D. J., and Viola, F. (2018). Taming VAEs. *arXiv:1810.00597 [cs, stat].* doi: 10.48550/arXiv.1810.00597

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). "High-resolution image synthesis with latent diffusion models," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (New Orleans, LA: IEEE).

Ronneberger, O., Fischer, P., and Brox, T. (2015). "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2015*, eds N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi (Cham: Springer International Publishing), 234–241.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature* 323, 533–536. doi: 10.1038/323533a0

Salehi, P., Chalechale, A., and Taghizadeh, M. (2020). Generative adversarial networks (GANs): an overview of theoretical model, evaluation metrics, and recent developments. *arXiv:2005.13178 [cs, eess].* doi: 10.48550/arXiv.2005.13178

Salvatori, T., Song, Y., Lukasiewicz, T., Bogacz, R., and Xu, Z. (2021). Predictive coding can do exact backpropagation on convolutional and recurrent neural networks. *arXiv:2103.03725 [cs].* doi: 10.48550/arXiv.2103.03725

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120 [cond-mat, q-bio, stat].* doi: 10.48550/arXiv.1312.6120

Schirrmeister, R. T., Chrabaszcz, P., Hutter, F., and Ball, T. (2018). Training generative reversible networks. *arXiv:1806.01610 [cs, stat].* doi: 10.48550/arXiv.1806.01610

Seitzer, M. (2020). pytorch-fid: FID score for PyTorch. *Version 0.1.1.*

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2020). Grad-CAM: visual explanations from deep networks via gradient-based localization. *Int. J. Comput. Vis.* 128, 336–359. doi: 10.1007/s11263-019-01228-7

Shahroudnejad, A. (2021). A survey on understanding, visualizations, and explanation of deep neural networks. *arXiv:2102.01792 [cs]*. doi: 10.48550/arXiv.2102.01792

Shen, Y., Gu, J., Tang, X., and Zhou, B. (2020). "Interpreting the latent space of gans for semantic face editing," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Seattle, WA), 9240–9249.

Sikka, H., Zhong, W., Yin, J., and Pehlevan, C. (2019). A Closer look at disentangling in β-VAE. *arXiv:1912.05127 [cs, stat]*. doi: 10.1109/IEEECONF44664.2019.9048921

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research*, eds F. Bach and D. Blei (Lille: PMLR), 2256–2265.

Sohoni, N. S., Aberger, C. R., Leszczynski, M., Zhang, J., and Ré, C. (2019). Low-memory neural network training: a technical report. *arXiv:1904.10631 [cs, stat]*. doi: 10.48550/arXiv.1904.10631

Song, Y., Lukasiewicz, T., Xu, Z., and Bogacz, R. (2020). Can the brain do backpropagation? exact implementation of backpropagation in predictive coding networks. *Adv. Neural Inf. Process. Syst.* 33, 22566–22579.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2015). Striving for simplicity: the all convolutional net. *arXiv:1412.6806 [cs]*. doi: 10.48550/arXiv.1412.6806

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV: IEEE), 2818–2826.

Takida, Y., Liao, W.-H., Uesaka, T., Takahashi, S., and Mitsufuji, Y. (2021). Preventing posterior collapse induced by oversmoothing in gaussian VAE. *arXiv:2102.08663 [cs]*. doi: 10.48550/arXiv.2102.08663

Thakur, N., and Han, C. Y. (2021). A study of fall detection in assisted living: identifying and improving the optimal machine learning method. *J. Sensor Actuator Netw*. 10, 39. doi: 10.3390/jsan10030039

Velichko, A. (2020). Neural network for low-memory IoT devices and MNIST image recognition using kernels based on logistic map. *Electronics* 9, 1432. doi: 10.3390/electronics 9091432

Wang, Y., Ni, Z., Song, S., Yang, L., and Huang, G. (2021). "Revisiting locally supervised learning: an alternative to end-to-end training," in *International Conference on Learning Representations* (Virtual Conference).

Whittington, J. C., and Bogacz, R. (2019). Theories of error back-propagation in the brain. *Trends Cogn. Sci.* 23, 235–250. doi: 10.1016/j.tics.2018.12.005

You, Y., Gitman, I., and Ginsburg, B. (2017). Large Batch training of convolutional networks. *arXiv:1708.03888 [cs]*.

Zintgraf, L. M., Cohen, T. S., and Welling, M. (2017). A new method to visualize deep neural networks. *arXiv:1603.02518 [cs]*. doi: 10.48550/arXiv.1708.03888