Feature Alignment for Approximated Reversibility in Neural Networks

Tiago de Souza Farias
Jonas Maziero

Departamento de Física, Centro de Ciências Naturais e Exatas
Universidade Federal de Santa Maria
Avenida Roraima 1000, Santa Maria, Rio Grande do Sul, 97105-900, Brazil

Abstract

We introduce feature alignment\(^1\), a technique for obtaining approximate reversibility in artificial neural networks. By means of feature extraction, we can train a neural network to learn an estimated map for its reverse process from outputs to inputs. Combined with variational autoencoders, we can generate new samples from the same statistics as the training data. Improvements of the results are obtained by using concepts from generative adversarial networks. Finally, we show that the technique can be modified for training neural networks locally, saving computational memory resources. Applying these techniques, we report results for three vision generative tasks: MNIST, CIFAR-10, and celebA.

Keywords: Neural networks, features, reversibility, local training

1. Introduction

Feature extraction (Olah et al., 2017) is a set of techniques aiming to find inputs, known as features, that maximize the activation of one or more neurons. Usually, feature extraction is used as a method for model interpretability, where one seeks to understand a neural network by analyzing how much each neuron contributes for modelling the training data. The process of obtaining these features is, in a sense, an attempt towards reversing a neural network. Since a neural network is composed by functions which map inputs to outputs, a feature is the input we would have given a target activation for a group of posterior selected neurons.

Reversibility of neural networks relates to how well one can reverse the map from the activation of target neurons to the input neurons (Aidan N. Gomez and Grosse, 2017). Usually, neural networks are not reversible, due to a few reasons: non-reversible activation functions, for example ReLU (Agarap, 2019), non-orthogonal weights, and lack of one-to-one relationships. One advantage of reversible neural networks, besides the reversing mapping, is its memory efficiency: while non-invertible networks need to store all the activations for the backward pass during training, reversible networks allows the storage of fewer activations to update the trainable parameters.

Reversibility also constrains the number of possible models, as there are many possible parameters configurations which model the data. For example, if one considers an analytical function that one wishes to approximate by a sufficiently parameterized neural network, with the pair of data \(\{x, f(x)\}\), there are a number of local minima which estimates the function\(^1\).

\(^1\) Code available at https://github.com/tiago939/feature_alignment
\[ x \to f(x) \], each one obtained by a different random initialization of the neural network parameters (assuming optimal convergence). By imposing the restriction of reversibility \( f(x) \to x \), we can reduce the number of optimal points toward which a neural network can converge.

Memory is often a bottleneck for neural networks. The backpropagation algorithm (Linnaimmaa, 1976; Rumelhart et al., 1986), widely used in modern deep learning for training, requires the storage of the activation of all neurons of a network in order to update its parameters. Local training rules (Baldi and Sadowski, 2016) allow a more efficient memory optimization of neural networks. 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 save memory resources and has many potential applications, from low-memory devices (Velichko, 2020; Sohoni et al., 2019) to training large batch sizes (Gao and Zhong, 2020; You et al., 2017), and, even further, to train very large neural networks (Jing and Xu, 2019).

This paper makes use of feature extraction for inverse mapping, a technique for approximated reversibility of neural networks that can be implemented in several architectures. We show that feature alignment makes neural networks approximately reversible for a given training data. We also adapt the technique for local training, showing results for the reconstruction of images.

2. Related work

Several works have been done on the area of feature extraction, specially applied for model interpretability and explainability (Shahroudnejad, 2021; Fan et al., 2021; Gilpin et al., 2019). These techniques, used for extracting features, usually consist in activation maximization (Mahendran and Vedaldi, 2016), 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 maximising its activation.

Many techniques of feature extraction consist in studying features in already pretrained classifiers (Nguyen et al., 2016b). Other techniques consist in searching for features in the latent space (Yujun Shen and Zhou, 2020). Feature extraction can also be used for understanding which parts of an input contribute the most for the target activations (Selvaraju et al., 2020; Springenberg et al., 2015; Zintgraf et al., 2017).

In a generative process, we want to produce new examples with the same statistical distribution as the training data. There are different techniques to model the data for generation. Among these techniques, autoencoders, generative adversarial networks, and normalizing flows are very popular. Autoencoders (AE) have two networks: an encoder that projects the inputs into a vector, usually with 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 a unsupervised fashion, nevertheless there are variants that include labeled information to further increase training for a specific objective. Variational autoencoders (VAE) (Kingma and Welling, 2014, 2019; Doersch, 2021) improve the sampling ability by projecting the data into a prob-
abilistic latent vector. Generative adversarial networks (GANs) (Goodfellow et al., 2014; Salehi et al., 2020; Gui et al., 2020) are another example of a generative method. By having two competing networks, a generative which takes a random low-dimensional input and outputs an image, and a discriminator network that compares the images from the training data set 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 data set as possible. Normalizing flows (Papamakarios 2019; Kobyzev et al. 2020; Kingma and Dhariwal, 2018) generate images by transforming a simple distribution to a more complex one by a series of reversible transformations.

There have been works combining autoencoders with GANs (Larsen et al., 2016). The work done in Refs. Nguyen et al. (2016a); Dosovitskiy and Brox (2016) is related to ours. They synthesized new images with the same statistics as the training data by imputing 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 (Atapattu and Rekabdar, 2019. Schirrmeister et al., 2018; Grathwohl et al., 2018; Behrmann et al., 2019; Leemon Baird and Ingkiriwang, 2005). 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 data set directly.

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). 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 (Millidge et al., 2020; Salvatori et al., 2021; Lillicrap et al., 2020).

3. Methods

Let $E(x; \theta)$ be a neural network with inputs $x$ and trainable parameters $\theta$. The output is a latent vector $z_x$ with an arbitrary number of neurons.

3.1 Feature alignment

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

The feature $\hat{r}$ of $z(x)$ is obtained by minimizing a distance function $\mathcal{L}(z_x, z_r)$ with respect to the random inputs $r$. By minimizing this distance, the random input will evolve continually according to a gradient flow:

$$\frac{\partial r}{\partial t} = -\frac{\partial \mathcal{L}}{\partial r}.$$  \hspace{1cm} (1)
In feature alignment, we have two encoders that share the same parameters. Left: at training time, the network is trained to approximate $\hat{r} \approx x$. Right: during inference, we sample $z_x$ from either the data or a known distribution to obtain the features.

Since $z(x)$ is fixed, $r$ will evolve such as the function of the random variables will approximate $z(x)$ as much as possible (See appendix A.1). 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:

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

with $\tau$ 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 activation. These updates are done in $T$ time steps. Properly optimized, the solution to equation (2) converges to a feature by approximating the inverse of the weights (see Appendix 1). So then, optimizing the parameters of the network, the weight matrix between layers will have the orthogonal property $w^T w = I$, which implies in approximated reversibility (see Appendix 2).

After we extract the feature $\hat{r}$, we measure how similar it is to the inputs $x$ by a new loss function $L(x, \hat{r})$. 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. Notice that we initialize $r$ with the same shape as $x$.

### 3.2 Variational autoencoders with feature alignment (VFA)

One issue with the encoder, and autoencoders in general, is its inability to generate new samples with the same statistical distribution as the training data. The latent variables from the data are associated to a distribution of variables which might be too complicated for an effective sampling. To solve this problem, we use a variational autoencoder (VAE) formulation, except by removing the decoder network. In the VAE, the output of the encoder is coupled with two layers that return the mean value $\mu_x$ and variance $\sigma^2_x$ of the
data. We constrain the latent vector to have a distribution that is easy to sample (usually a Gaussian distribution), by comparing two probability distributions via a metric such as the Kullback-Leibler divergence. So, the cost function:

$$ \mathcal{L} = ||x - \hat{r}||^2_2 - \beta D_{KL}(q_\theta(z|x)||p(z)),$$

is used to train the feature encoder with a constraint to the output latent variables from a known random probability distribution $p(z)$, from which we can easily sample. The constant $\beta$ is a hyper-parameter that regularizes the latent vector for better disentanglement representation of the data (Higgins et al., 2016; Burgess et al., 2018; Sikka et al., 2019).

We choose the distribution $p(z)$ according to the principle of maximum entropy: since the latent variables are in the range $(-\infty, +\infty)$, the Gaussian distribution is best suited in this case. We then constrain each latent variable as to come from a Gaussian distribution with zero mean value and variance equal to one. Just as the VAE, we can not train the encoder by sampling directly from the mean and variance of the latent vector. Instead, we use the renormalization trick: we sample a random vector $\epsilon$ from a normal distribution, the latent vector is then: $z_x = \mu_x + \epsilon \odot \sigma^2_x$, with $\odot$ the element-wise product. Note that we do not have an random normal vector for $z_r$, since its purpose is solely to reconstruct $x$.

Figure 3 summarizes training a VAE with feature alignment.

### 3.3 Improving the quality of the features

As will be shown in the results section, the features extracted using the feature alignment trained with VAE are blurry, and this is due to the variational autoencoder nature (Rezende and Viola, 2018). To improve the quality of the generated images, we couple a generator network $G$ and a discriminator network $D$ to the features. In this way, the feature acts as a second latent vector, with a more complex distribution sampled from the simpler one. This generator network is similar to the refiner network presented in Ref. Atapattu and Rekabdar (2019), which take an image as input and outputs an improved version of it.

The generator takes the feature as input and produces a new output which is compared to the input $x$. The discriminator is trained as a generative adversarial network, evaluating the
probability that $G(\hat{r})$ is real or fake. 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, we can use the Wasserstein GAN formulation (Arjovsky et al., 2017), which replaces the discriminator by a critic network that measures the score of the “realness” of an image.

To reduce possible effects due to the posterior collapse problem in VAEs (Takida et al., 2021; Havrylov and Titov, 2020; Lucas et al., 2019) and to balance with the reconstruction loss, we propose a random schedule for the $\beta$ variable. For each example during training, we sample $\beta$ from a uniform distribution $U(0, 1)$.

While the reconstruction of images is optimized usually with pixel level loss, we can also consider high level properties of the data. This kind of measure, called perceptual loss (Johnson et al., 2016), compares the output of the reconstruction with the original image at high level neurons (presented near the end of the network). Here we consider the perceptual loss using the mean and variance layers from the encoder network, enforcing the reconstruction to have the same statistical properties of the original input.

Figure 2: Variational autoencoder with feature alignment. Left: training the encoder to reconstruct the inputs $x$. Right: we sample a random normal vector $\epsilon$ to generate new data.

Figure 3: Training in the VFA setting, with the addition of generator and discriminator networks.
The final losses, for the encoder, generator, and discriminator are, respectively:

\[
\mathcal{L}_E = \|x - \hat{r}\|^2 + \beta \mathcal{D}_{KL}(q_{\theta}(z|x)\|p(z)), \tag{4}
\]

\[
\mathcal{L}_G = \|1 - D(G(\hat{r}))\|^2 + \lambda (\|\mu(x) - \mu(G(\hat{r}))\|^2 + \|\sigma^2(x) - \sigma^2(G(\hat{r}))\|^2), \tag{5}
\]

\[
\mathcal{L}_D = \|1 - D(x)\|^2 + \|1 - D(G(\hat{r}))\|^2. \tag{6}
\]

with \(\lambda\) a hyperparameter that weights the perceptual loss contribution.

If a training data has additional information, such as labels, we can train a neural network jointly with supervised training for specific related tasks such as conditional generation, where we want samples that 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 from the Gaussian distribution \(z \sim Q(z_i, c_i)\), with \(c_i\) being a one-hot vector containing the class information. Similar to the work done in Ref. 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, at inference time we can choose a greater value for the one-hot vector, which puts emphasis on the chosen class.

### 3.4 Local feature alignment

The rules for feature alignment were presented as a global rule: the auxiliary loss is defined with the output layer and the loss is defined with the input layer, so we have full communication between all layers. However, we can reformulate this rule with local losses, similar to target propagation rules (Farias and Maziero, 2018; Bengio, 2014; 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 from the first to the last, we activate it from the its inputs and store a second activation from a random input with the same dimension, we then extract the feature by optimizing the random input with an auxiliary loss between activation of the random output and the true input. Finally, the parameters of the chosen layer is updated by optimizing the loss between the reconstruction and true input. This technique of local training is summarized in the Algorithm 2 and illustrated at FIG. 4.

The neural network becomes a predictive machine, where each layer trains its parameters by trying to predict the inputs by knowing the outputs. The constraint of the local learning has a more pronounced effect on the non-linearity of a neural network trained in this way. While backpropagation can adjust all the parameters of a network so that the feature reconstructs the input, local rules can only rely on very strict information content available. Non reversible functions, such as the ReLU function, propagate loss of information, which will lead to reconstructions with low fidelity. Thus, a non-linear function must be carefully chosen in order to preserve as much information as possible. The function inverse hyperbolic sine \((\text{arsinh}(x) = \ln(x + \sqrt{1 + x^2}))\), 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 \(\tanh\). These properties make \(\text{arsinh}\) a good candidate function for local training.
Algorithm 2 Training with local feature alignment

1: for l=0, L do $\triangleright$ for each layer
2: $z^l_x = E(x; \theta_l)$
3: initialize $r = x.\text{shape}$
4: $t = 0$
5: while $t < T$ do
6: $z^l_r = E(r; \theta_l)$
7: $L_l = ||z^l_x - z^l_r||^2$
8: $r = r - \tau \frac{\partial L_l}{\partial r}$
9: $t = t + 1$
10: end while
11: $\hat{r} = r$
12: $C_l = ||x - \hat{r}||^2$
13: update $\theta_l$ by optimizing $C_l$
14: $x = z^l_x.\text{detach}$
15: end for

Figure 4: The local training rule.

At inference time, we need to propagate the information backwards from the output to the input layer by layer just as one would do normally with non-local feature extraction. However, the non-linear function will play a major role here, it being reversible will be required to approximate reversibility. This is done by, after inputting the latent vector, applying the inverse of the non-linear function after each layer that uses the function. This process is summarized in the Algorithm 3.

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
Algorithm 3: Reconstruction with local feature alignment

1: sample $z^{l_f}_x$
2: for $l=L, 0$ do
3: \hspace{1em} initialize $r = x$.shape with $x$ as $z^{l_f}_x = E(x; \theta_l)$
4: \hspace{1em} $t = 0$
5: \hspace{1em} while $t < T$ do
6: \hspace{2em} $z^{l_f}_r = E(r; \theta_l)$
7: \hspace{2em} $L_l = ||z^{l_f}_x - z^{l_f}_r||_2^2$
8: \hspace{2em} $r = r - \tau \frac{\partial L_l}{\partial r}$
9: \hspace{2em} $t = t + 1$
10: \hspace{1em} end while
11: \hspace{1em} $\hat{r} = r$
12: \hspace{1em} if layer = non-linear function $f$ then
13: \hspace{2em} $\hat{r} = f^{-1}(\hat{r})$
14: \hspace{1em} end if
15: \hspace{1em} $z^{l_f}_x = \hat{r}$
16: end for

the last layer that outputs a single value for each example. Only the generator uses batch normalization after each convolution. All convolutions have kernel size $k = 3$.

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 $\beta$ from a uniform distribution, with a different random value for each training example.

From Appendix 2, the loss become 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, i.e.:

$$w = \begin{cases} -\sqrt{2} & \text{if } w < -\sqrt{2}, \\ \sqrt{2} & \text{if } w > \sqrt{2}, \\ w & \text{otherwise}. \end{cases} \quad (7)$$

For the image data sets, the generator is composed by a few convolutional layers, with same number of filters and kernel sizes.

We also report the results for the modified feature alignment for local training as a proof of concept by training a 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 $\arg \min_z ||x - G(z)||_2^2$.

5. Results

We compare the results against traditional variational autoencoders and generative adversarial networks. The results show reconstruction of the inputs with feature and generator. We also show random samples from the generator network. We evaluate the Fréchet Inception Distance (FID) (Heusel et al., 2018; Seitzer, 2020) for each data set as a measure of
the quality of the samples. 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 data set (here we choose 10k images) and sampled from a generator model:

\[
FID = \|\mu_1 - \mu_2\|^2_2 + tr(\Sigma_1 + \Sigma_2 - 2\sqrt{\Sigma_1 \Sigma_2}),
\]

with \(\mu\) the mean of activations, \(\Sigma\) the covariance matrix and \(tr\) the trace function. The results are shown in Table 1.

| Method | MNIST  | CelebA | CIFAR-10 |
|--------|--------|--------|----------|
| VAE    | 29.99  | 82.93  | 164.62   |
| GAN    | 18.72  | 31.40  | 63.36    |
| \(\hat{r}\) | 129.82 | 122.47 | 205.65   |
| \(G(\hat{r})\) | 38.53  | 128.35 | 69.89    |

Table 1: FID scores across three image data sets.

5.1 MNIST

The MNIST data set (Bengio and Haffner, 1998) is a set of 60000 grayscale images with size 28 × 28 pixels that contains hand drawings of digits from zero to nine. Figure 5 shows the reconstruction of images by the features and with the generator applied to them, compared with traditional AE, VAE and GAN. While the features tend to be noisy, the generator can sharpen the images to resemble better the original inputs. Figure 6 shows random samples from the generator with a class layer coupled to the encoder. The same image also shows the interpolation of the latent vector between four pairs of images to demonstrate the continuity of transitions on the latent space.

![Figure 5: Reconstruction of images of the MNIST data set from four different models.](image-url)
Figure 6: Left: random samples. Right: interpolation among four images reconstructed from the data set.

5.2 CelebA

The CelebA data set (Liu et al., 2015) is a set of 202600 images of celebrity faces. We resize the images to $64 \times 64$ pixels. Figures 7 and 8 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.

Figure 7: Reconstruction of images of the CelebA data set from four different models.
5.3 CIFAR-10

The CIFAR-10 data set (Krizhevsky and Hinton, 2009) contains 70000 natural images with size $32 \times 32$ pixels across 10 different classes. Figure 9 shows the reconstruction of images from the data set. While the features do approximate the original inputs, the transformation of the generator tends to be more dissimilar due its loss being dependent only on the adversarial contribution ($\lambda = 0$). Just as before, Figure 10 shows random samples and interpolation, which show a diversity of images, albeit less perceptual similar to the original data set.

Figure 8: Two sets of interpolation among four random sampled images.

Figure 9: Reconstruction of images on CIFAR-10.
It is important to note that we do not expect feature alignment to exceed the reconstruction and sample qualities of VAEs and GANs. The condition of reversibility constrains the optimization of a neural network and thus has to balance the reversibility cost with other losses. Nevertheless we compare the reconstruction $L_2$ loss with other networks to analyse how different each network is compared to same metric. These results are shown in FIG. 11.

As expected, autoencoders have the lowest loss, given that it is optimized directly for reconstruction. The features should be compared to VAEs and the generator with GAN, since they are optimized with similar losses. Here we can see that absence of perceptual loss on the generator network decreases the reconstruction (without optimizing the latent vector) ability.
5.4 Local feature alignment

Here we show results from an encoder trained for reconstruction with the local feature alignment training. Figures 12, 13, and 14 show reconstruction pairs for the MNIST, CIFAR-10, and CelebA data sets respectively. We can see that local training can reconstruct images despite the layers not receiving information from the reconstruction loss of images, which can be explained with the same reason of non-local feature alignment: the weights form a orthogonal matrix which tries to reverse information between layers as much as possible, which is limited only by the network capacity, directly related to the number of neurons.

Figure 12: Local feature alignment on the MNIST data set. Each pair of images contains the reconstruction and original, respectively.

Figure 13: Local feature alignment on the CIFAR-10 data set. Each pair of images contains the reconstruction and original, respectively.

Figure 14: Local feature alignment on the CelebA data set. Each pair of images contains the reconstruction and original, respectively.
6. Conclusions

We presented feature alignment, a technique that can approximately reverse neural networks. We trained an encoder to predict its own input by optimizing the features to match the inputs. By coupling a probabilistic layer with the same formulation as the variational autoencoders, we have the ability to generate new samples with the same statistical distribution of the training data. To improve the quality of the generated samples, that suffer from noise effects, we combined the generative adversarial network method by coupling a generator and a discriminator network to the features. We also showed that the technique can be modified to a local training rule, as opposed to backpropagation, which offers the advantage of lowering memory resources for training and extracting gradients of 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 own inputs. The restriction is the network’s architecture itself, since the bottlenecks do not allow for a one-to-one relationship.

The results show that the features can approximately match the inputs. Despite not being able to improve on the sampling quality of other current generative techniques, reversibility can offer advantages when a mapping of the outputs back to its inputs is desired. The data set CIFAR-10 is notoriously difficult because of the small image size and high variance, which led to samples with high FID measure. By coupling a classification layer on the encoder network, we can use label information for conditional generation of samples. Additionally, the results of local training suggests that we can train neural networks without feedback of global loss functions.

The proposed technique can be seen as a compromise between VAEs and GANs. The full architecture has crisper samples than VAEs and a latent vector that can be exploited. Feature extraction can be implemented across many different neural network architectures. Thus, it can potentially be improved to further increase the quality of the results.

Appendix A. Mathematical analysis

This section studies the numerical analysis of the technique.

A.1 Convergence of the features

For two fully connected layers, we have $a_j^{(x)} = \hat{a}_j = \sum_i w_{ij} x_i$ and $a_j^{(r)} = \sum_i w_{ij} r_i$. The feature is obtained by optimizing the squared $L_2$ loss between the two activations:

$$\mathcal{L} = \frac{1}{2}(\hat{a}_j - a_j^{(r)})^2,$$

which follows that $r$ will evolve with the gradient flux:

$$\frac{\partial r_t}{\partial t} = -\frac{\partial \mathcal{L}}{\partial r_t} \implies r_t = r_{t-1} - \frac{\partial C}{\partial r_{t-1}}.$$
Then we can evaluate $r^t$ at each time step $t$:

\begin{align}
  r^1_i &= r^0_i - w_{ij} (\hat{a}_j - w_{ij} r^0_i) = r^0_i (1 - w^2_{ij}) + w_{ij} \hat{a}_j, \\
  r^2_i &= r^1_i (1 - w^2_{ij}) + w_{ij} \hat{a}_j = r^0_i (1 - w^2_{ij})^2 + [1 + (1 - w^2_{ij})] w_{ij} \hat{a}_j, \\
  r^3_i &= r^0_i (1 - w^2_{ij})^3 + [1 + (1 - w^2_{ij}) + (1 - w^2_{ij})^2] w_{ij} \hat{a}_j, \\
  r^4_i &= r^0_i (1 - w^2_{ij})^4 + [1 + (1 - w^2_{ij}) + (1 - w^2_{ij})^2 + (1 - w^2_{ij})^3] w_{ij} \hat{a}_j.
\end{align}

With the pattern above, we generalize $r^t$ for any time step:

\[ r^p_i = r^0_i (1 - w^2_{ij})^p + \sum_{q=0}^{p-1} (1 - w^2_{ij})^q w_{ij} \hat{a}_j. \]  

(15)

Under the restriction of $w^2_{ij} \leq 2$, we have at limit as $t$ grows:

\[ \lim_{p \to \infty} r^0_i (1 - w^2_{ij})^p = 0 \quad \therefore \quad \lim_{p \to \infty} \sum_{q=0}^{p-1} (1 - w^2_{ij})^q w_{ij} \hat{a}_j = \frac{\hat{a}_j}{w_{ij}}. \]  

(16)

So the loss $\mathcal{L} \to 0$ as $p \to \infty$.

A.2 Convergence of the weights

The $L_2$ loss that update the parameters is:

\[ \mathcal{L} = \frac{1}{2} (x_i - r^t_i)^2 = \frac{1}{2} \left[ x_i - r^0_i (1 - w^2_{ij})^t - \sum_{q=0}^{t-1} (1 - w^2_{ij})^q w_{ij} \hat{a}_j \right]^2. \]  

(17)

For one-shot, $T = 1$, training, we have:

\[ \mathcal{L} = \frac{1}{2} \left[ x_i - r^0_i (1 - w^2_{ij}) - w_{ij} \hat{a}_j \right]^2 = \frac{1}{2} \left[ x_i - r^0_i (1 - w^2_{ij}) - w_{ij} x_i \right]^2. \]  

(18)

We can rewrite the equation above in vector notation as follows:

\[ \mathcal{L} = \frac{1}{2} \left[ x - r^0 (I - w^T w) - w^T w x \right]^2. \]  

(19)

For any $r^0$, the equation above has roots for $w^2_{ii} = 0$. We can see then that the loss is minimal when the weight matrix product is orthogonal, i.e. $w^T w = I$, which has the consequence that the transposed weight matrix is also its generalized Moore–Penrose inverse or pseudo-inverse $w^T = w^{-1}$.

References

Abien Fred Agarap. Deep Learning using Rectified Linear Units (ReLU). arXiv:1803.08375, February 2019.

Raquel Urtasun Aidan N. Gomez, Mengye Ren and Roger B. Grosse. The reversible residual network: Backpropagation without storing activations. arXiv:1707.04585, 2017.
Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W. Pellegrini, Ralf S. Klessen, Lena Maier-Hein, Carsten Rother, and Ulrich Köthe. Analyzing Inverse Problems with Invertible Neural Networks. arXiv:1808.04730, February 2019.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. arXiv:1701.07875, December 2017.

Charith Atapattu and Banafsheh Rekabdar. Improving the realism of synthetic images through a combination of adversarial and perceptual losses. In 2019 International Joint Conference on Neural Networks (IJCNN), pages 1–7, July 2019. doi: 10.1109/IJCNN.2019.8852449. ISSN: 2161-4407.

Pierre Baldi and Peter Sadowski. A Theory of Local Learning, the Learning Channel, and the Optimality of Backpropagation. Neural Networks, 83:51–74, November 2016. ISSN 08936080. doi: 10.1016/j.neunet.2016.07.006. arXiv: 1506.06472.

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible Residual Networks. arXiv:1811.00995, May 2019.

Yann LeCun Leon Bottou Yoshua Bengio and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, November 1998. ISSN 1558-2256. doi: 10.1109/5.726791.

Yoshua Bengio. How Auto-Encoders Could Provide Credit Assignment in Deep Networks via Target Propagation. arXiv:1407.7906, September 2014.

Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae. arXiv:1804.03599, April 2018.

Carl Doersch. Tutorial on Variational Autoencoders. arXiv:1606.05908, January 2021.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial Feature Learning. arXiv:1605.09782, April 2017.

Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. arXiv:1602.02644, feb 2016.

Fenglei Fan, Jinjun Xiong, Mengzhou Li, and Ge Wang. On Interpretability of Artificial Neural Networks: A Survey. arXiv:2001.02522, January 2021.

Tiago de Souza Farias and Jonas Maziero. Gradient target propagation. arXiv:1810.09284, November 2018.

Fengli Gao and Huicai Zhong. Study on the Large Batch Size Training of Neural Networks Based on the Second Order Gradient. arXiv:2012.08795, December 2020.

Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining Explanations: An Overview of Interpretability of Machine Learning. arXiv:1806.00069, February 2019.
Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. arXiv:1406.2661, June 2014.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. arXiv:1810.01367, October 2018.

Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. arXiv:2001.06937, January 2020.

Serhii Havrylov and Ivan Titov. Preventing Posterior Collapse with Levenshtein Variational Autoencoder. arXiv:2004.14758, April 2020.

Donald Olding Hebb. The organization of behavior; a neuropsychological theory. New York: Wiley, 1949. doi: 10.1002/cne.900930310.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. arXiv:1706.08500, January 2018.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. November 2016.

Wei Hu, Lechao Xiao, and Jeffrey Pennington. Provable Benefit of Orthogonal Initialization in Optimizing Deep Linear Networks. arXiv:2001.05992, January 2020.

Takuya Isomura and Taro Toyoizumi. A Local Learning Rule for Independent Component Analysis. Scientific Reports, 6(1):28073, June 2016. ISSN 2045-2322. doi: 10.1038/srep28073.

Takuya Isomura and Taro Toyoizumi. Error-Gated Hebbian Rule: A Local Learning Rule for Principal and Independent Component Analysis. Scientific Reports, 8(1):1835, January 2018. ISSN 2045-2322. doi: 10.1038/s41598-018-20082-0.

Kun Jing and Jungang Xu. A Survey on Neural Network Language Models. arXiv:1906.03591, June 2019.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. arXiv:1603.08155, March 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. arXiv:1412.6980, January 2017.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. arXiv:1807.03039, July 2018.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. arXiv:1312.6114, May 2014.
Diederik P. Kingma and Max Welling. An Introduction to Variational Autoencoders. arXiv:1906.02691, December 2019. doi: 10.1561/2200000056.

Ivan Kobyzev, Simon J. D. Prince, and Marcus A. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. arXiv:1908.09257, June 2020. doi: 10.1109/TPAMI.2020.2992934.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Master’s thesis, Department of Computer Science, University of Toronto, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems, 25, 2012.

Dmitry Krotov and John J. Hopfield. Unsupervised learning by competing hidden units. Proceedings of the National Academy of Sciences, 116(16):7723–7731, April 2019. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1820458116.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. arXiv:1512.09300, February 2016.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Unsupervised learning of hierarchical representations with convolutional deep belief networks. Communications of the ACM, 54(10):95–103, October 2011. ISSN 0001-0782. doi: 10.1145/2001269.2001295.

David Smalenberger Leemon Baird and Shawn Ingkiriwang. One-step neural network inversion with PDF learning and emulation. In Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005., volume 2, pages 966–971 vol. 2, July 2005. doi: 10.1109/IJCNN.2005.1555983. ISSN: 2161-4407.

Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. Backpropagation and the brain. Nature Reviews Neuroscience, 21(6):335–346, June 2020. ISSN 1471-0048. doi: 10.1038/s41583-020-0277-3.

Jack Lindsey and AshokLitwin-Kumar. Learning to Learn with Feedback and Local Plasticity. arXiv:2006.09549, June 2020.

Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. BIT Numerical Mathematics, 16(2):146–160, June 1976. ISSN 1572-9125. doi: 10.1007/BF01931367.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), December 2015.

James Lucas, George Tucker, Roger Grosse, and Mohammad Norouzi. Don’t Blame the ELBO! A Linear VAE Perspective on Posterior Collapse. arXiv:1911.02469, November 2019.
Aravindh Mahendran and Andrea Vedaldi. Visualizing Deep Convolutional Neural Networks Using Natural Pre-Images. *International Journal of Computer Vision*, 120(3):233–255, December 2016. ISSN 0920-5691, 1573-1405. doi: 10.1007/s11263-016-0911-8. arXiv: 1512.02017.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least Squares Generative Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821, October 2017. doi: 10.1109/ICCV.2017.304. ISSN: 2380-7504.

Beren Millidge, Alexander Tschantz, Anil K. Seth, and Christopher L. Buckley. Activation Relaxation: A Local Dynamical Approximation to Backpropagation in the Brain. *arXiv:2009.05359*, October 2020.

Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *arXiv:1605.09304*, November 2016a.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks. *arXiv:1602.03616*, May 2016b.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, November 2017. ISSN 2476-0757. doi: 10.23915/distill.00007.

Alexander G. Ororbia, Ankur Mali, Daniel Kifer, and C. Lee Giles. Conducting Credit Assignment by Aligning Local Representations. *arXiv:1803.01834*, July 2018.

George Papamakarios. Neural Density Estimation and Likelihood-free Inference. *arXiv:1910.13233*, October 2019.

Danilo Jimenez Rezende and Fabio Viola. Taming VAEs. *arXiv:1810.00597*, October 2018.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. ISSN 1476-4687. doi: 10.1038/323533a0.

Pegah Salehi, Abdolah Chalechale, and Maryam Taghizadeh. Generative Adversarial Networks (GANs): An Overview of Theoretical Model, Evaluation Metrics, and Recent Developments. *arXiv:2005.13178*, May 2020.

Tommaso Salvatori, Yuhang Song, Thomas Lukasiewicz, Rafal Bogacz, and Zhenghua Xu. Predictive Coding Can Do Exact Backpropagation on Convolutional and Recurrent Neural Networks. *arXiv:2103.03725*, March 2021.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120*, February 2014.

Robin Tibor Schirrmeister, Patryk Chrabaszcz, Frank Hutter, and Tonio Ball. Training Generative Reversible Networks. *arXiv:1806.01610*, August 2018.
Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid. August 2020. Version 0.1.1.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. International Journal of Computer Vision, 128(2):336–359, February 2020. ISSN 0920-5691, 1573-1405. doi: 10.1007/s11263-019-01228-7. arXiv: 1610.02391.

Atefeh Shahroudnejad. A Survey on Understanding, Visualizations, and Explanation of Deep Neural Networks. arXiv:2102.01792, February 2021.

Harshvardhan Sikka, Weishun Zhong, Jun Yin, and Cengiz Pehlevan. A Closer Look at Disentangling in $\beta$-VAE. arXiv:1912.05127, December 2019.

Nimit Sharad Sohoni, Christopher Richard Aberger, Megan Leszczyński, Jian Zhang, and Christopher Ré. Low-Memory Neural Network Training: A Technical Report. arXiv:1904.10631, April 2019.

Yuhang Song, Thomas Lukasiewicz, Zhenghua Xu, and Rafal Bogacz. Can the Brain Do Backpropagation? — Exact Implementation of Backpropagation in Predictive Coding Networks. Advances in Neural Information Processing Systems, 33:22566–22579, 2020.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. arXiv:1412.6806, April 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. pages 2818–2826, 2016.

Yuhta Takida, Wei-Hsiang Liao, Toshimitsu Uesaka, Shusuke Takahashi, and Yuki Mitsufuji. Preventing Posterior Collapse Induced by Oversmoothing in Gaussian VAE. arXiv:2102.08663, February 2021.

Andrei Velichko. Neural Network for Low-Memory IoT Devices and MNIST Image Recognition Using Kernels Based on Logistic Map. Electronics, 9(9):1432, September 2020. ISSN 2079-9292. doi: 10.3390/electronics9091432. arXiv: 2006.02824.

James C.R. Whittington and Rafal Bogacz. Theories of Error Back-Propagation in the Brain. Trends in Cognitive Sciences, 23(3):235–250, March 2019. ISSN 1364-6613. doi: 10.1016/j.tics.2018.12.005.

Yang You, Igor Gitman, and Boris Ginsburg. Large Batch Training of Convolutional Networks. arXiv:1708.03888, September 2017.

Xiaonan Tang Yujun Shen, Jinjin Gu and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, page 9240–9249, 2020. doi: 10.1109/CVPR42600.2020.00926.

Luisa M. Zintgraf, Taco S. Cohen, and Max Welling. A New Method to Visualize Deep Neural Networks. arXiv:1603.02518, June 2017.