Stabilizing GANs with Soft Octave Convolutions

Ricard Durall[1,2,3], Franz-Josef Pfreundt[1] and Janis Keuper[1,4]
[1]*Fraunhofer ITWM, Germany*
[2]*IWR, University of Heidelberg, Germany*
[3]*Fraunhofer Center Machine Learning, Germany*
[4]*Institute for Machine Learning and Analytics, Offenburg University, Germany*

Keywords: Generative Adversarial Network, Octave Convolution, Stability, Regularization.

Abstract: Motivated by recently published methods using frequency decompositions of convolutions (e.g. Octave Convolutions), we propose a novel convolution scheme to stabilize the training and reduce the likelihood of a mode collapse. The basic idea of our approach is to split convolutional filters into additive high and low frequency parts, while shifting weight updates from low to high during the training. Intuitively, this method forces GANs to learn low frequency coarse image structures before descending into fine (high frequency) details. We also show, that the use of the proposed soft octave convolutions reduces common artifacts in the frequency domain of generated images. Our approach is orthogonal and complementary to existing stabilization methods and can simply be plugged into any CNN based GAN architecture. Experiments on the CelebA dataset show the effectiveness of the proposed method.

1 INTRODUCTION

In recent years, unsupervised learning has received a lot of attention in computer vision applications. In particular, learning generative models from large and diverse datasets has been a very active area of research. Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) has risen as one of the main techniques that produces state-of-the-art results at generating realistic and sharp images. Unlike other generative methods (Kingma and Welling, 2013; Oord et al., 2016) that explicitly model maximum likelihood, GAN provides an attractive alternative that allows to model the density implicitly. Basically, it consists of training a generator and discriminator model in an adversarial game, such that the generator learns to produce samples from the data distribution. Nonetheless, despite their success, GANs often have an unstable training behaviour of which there is little to no theory explaining it. This makes it extremely hard to predict plausible results in new GAN experiments or to employ them in new domains. Consequently, their applicability is often drastically limited to a controlled and well-defined environment. In the literature, we encounter many current papers dedicated to finding heuristically stable architectures (Radford et al., 2015; Karras et al., 2017; Brock et al., 2018; Lin et al., 2019), loss functions (Mao et al., 2017; Arjovsky et al., 2017; Gulrajani et al., 2017) or regularization strategies (Miyato et al., 2018; Durall et al., 2020).

All the aforementioned generative methods have a common core building block: convolutions. In other words, they all are based on convolutional neural networks (CNN). That means that their architecture consist mostly of sets of stacked convolutional layers. Recent efforts have focused on improving these layers by reducing their inherent redundancy in density of parameters and in the amount of channel dimension of feature maps (Han et al., 2016; Luo et al., 2017; Chollet, 2017; Xie et al., 2017; Ke et al., 2017; Chen et al., 2019). These works analyse standard convolutional layers and their behaviour in detail. Basically, these layers are designed to detect local conjunctions of features from the previous layer and mapping their appearance to a feature map, which have always the same spatial resolution. However, natural images can be factorized into a low frequency signal that captures the global layout and coarse structure, and a high frequency part that captures fine details. Attracted by the idea of having feature maps with different resolutions and breaking with standard convolutional layers, some works (Ke et al., 2017; Chen et al., 2019) have built schemes, on top of standard CNNs architecture, that have access to different frequency content within the same feature map.

In this paper, we propose to replace standard convolutions from the architecture of GANs with novel soft octave convolutions. This replacement will have

Durall, R., Pfreundt, F. and Keuper, J.
Stabilizing GANs with Soft Octave Convolutions.
DOI: 10.5220/0010178700150023
In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 4: VISAPP*, pages 15-23
ISBN: 978-989-758-488-6

almost no impact on the architecture since octave convolutions are orthogonal and complementary to existing methods that also focus on improving CNN topology. We apply our model to the CelebA (Liu et al., 2015) dataset and demonstrate that, by simply substituting the convolutional layers, we can consistently improve the performances leading to a more stable training with less probability of mode collapses. Overall, our contributions are summarized as follows:

- We introduce a novel and generalizable convolution scheme for generative adversarial networks: the soft octave convolution.
- Our analysis shows that, using soft octave convolutions leads to more stable training runs and less frequency domain artifacts in generated images.
- We evaluate our approach by embedding the soft octave convolutions into different GAN architectures and provide both quantitative and qualitative results on the CelebA dataset.

2 RELATED WORK

Most of the deep learning approaches in computer vision are based on standard CNNs. They have heavily contributed in semantic image understanding tasks including the aforementioned works and references therein. In this work, we look at image generation techniques and we briefly review the seminal work in that direction. In particular, we focus our attention on a set of well-known GANs and the impact of alternative convolutional layers on these models.

2.1 Generative Adversarial Networks

The goal of generative models is to match real data distribution p_{data} with generated data distribution p_{g}. Thus, minimizing differences between two distributions is a crucial point for training generative models. Goodfellow *et al.* introduced an adversarial framework (GAN) (Goodfellow et al., 2014) which is capable of learning deep generative models by minimizing the Jensen-Shannon Divergence between p_{data} and p_{g}. This optimization problem can be described as a minmax game between the generator G, which learns how to generate samples which resemble real data, and a discriminator D, which learns to discriminate between real and fake data. Throughout this process, G indirectly learns how to model p_{data} by taking samples z from a fixed distribution p_z (e.g. Gaussian) and forcing the generated samples $G(z)$ to match p_{g}. The objective loss function is defined as

$$\min_G \max_D \mathcal{L}(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_z}[\log(1-D(G(\mathbf{z})))]. \quad (1)$$

Deep Convolutional GAN. Deep Convolutional GAN (DCGAN) (Radford et al., 2015) is one of the popular and successful network topology designs for GAN that in a certain way achieves a consistently stability during training. It is a direct extension of the GAN described above, except that it is mainly composed of convolution and transposed-convolution layers without max pooling or fully connected layers in both discriminator and generator.

Least-Squares GAN. Least-Squares GAN (LSGAN) (Mao et al., 2017) also tries to minimize Pearson X^2 divergence between the real and the generated distribution. The standard GAN uses a sigmoid cross entropy loss for the discriminator to determine whether its input comes from p_{data} and p_{g}. Nonetheless, this loss has an important drawback. Given a generated sample is classified as real by the discriminator, then there would be no apparent reason for the generator to be updated even though the generated sample is located far from the real data distribution. In other words, sigmoid cross entropy loss can barely push such generated samples towards real data distribution since its classification role has been achieved. Motivated by this phenomenon, LSGAN replaces a sigmoid cross entropy loss with a least square loss, which directly penalizes fake samples by moving them close to the real data distribution.

Wasserstein GAN. Wasserstein GAN (WGAN) (Arjovsky et al., 2017) suggests the Earth-Mover (EM) distance, which is also called the Wasserstein distance, as a measure of the discrepancy between the two distributions. The benefit of the EM distance over other metrics is that it is a more sensible objective function when learning distributions with the support of a low-dimensional manifold. EM distance is continuous and differentiable almost everywhere under Lipschitz condition, which standard feed-forward neural networks satisfy. In order to enforce such a condition, weight clipping is used on each neural network layer. Its main idea is to clamp the weight to a small range, so that the Lipschitz continuity is guaranteed. Finally, since EM distance is intractable, it is converted in to a tractable equation via Kantorovich-Rubinstein duality with the Lipschitz function.

2.2 Convolutional Layers

Standard convolutional layers are designed to detect local conjunctions of features from the previous layer and to project their appearance to a feature map which does not vary its spatial resolution at any time. Nevertheless, in accordance with the spatial-frequency model (Campbell and Robson, 1968; DeValois and DeValois, 1990), natural images can be factorized into a low frequency signal that captures the global layout and coarse structure, and a high frequency signal that captures fine details. Attracted by the idea of having feature maps with different resolution, recent works tried to leverage frequency decompositions for deep learning approaches. For example, (Ke et al., 2017; Chen et al., 2019) have built architectures on top of standard CNNs, that have access to different frequency content. (Ke et al., 2017) suggested a multigrid architecture, that has the intention of wiring cross-scale connections into network structure at the lowest level. In order to create such a topology, every convolutional filter extends spatially within grids (h,w), across grids multiple scales (s) within a pyramid, and over corresponding feature channels (c). Building in this fashion, a combination of pyramids across the architecture (h,w,s,c).

2.3 Octave Convolutions

The original approach towards octave convolutions has been introduced by (Chen et al., 2019). Given the input feature tensor of a convolutional layer $X \in \mathbb{R}^{c\times h\times w}$ with channels c and spacial resolutions in height h and width w, (Chen et al., 2019) suggested to factorize it along channel dimension into two groups, one for low frequencies and one for high frequencies $X = \{X^{\mathrm{H}}, X^{\mathrm{L}}\}$ (see Fig. 1 for details). The authors argued, that the subset of the feature maps that capture spatially low frequency changes contains spatially redundant information. In order to reduce the spatial redundancy, they introduced the octave feature representation, which corresponds to a division of the spatial dimensions by 2 for some of the feature maps.

In order to control the factorization into high- and low frequency parts, (Chen et al., 2019) introduced the hyper-parameter $\alpha \in [0,1]$ (see Fig. 2)

$$X^{\mathrm{L}} \in \mathbb{R}^{\alpha c\times \frac{h}{2}\times \frac{w}{2}} \quad \text{and} \quad X^{\mathrm{H}} \in \mathbb{R}^{(1-\alpha)c\times h\times w}. \qquad (2)$$

However, this formulation has a major drawback in practice: α regulates the frequency decomposition at an architectural level. Changing α causes the network topology to change and thus can not be done during training.

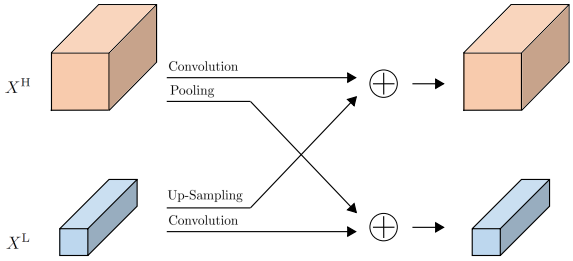


Figure 1: Original formulation of the octave convolution as introduced by (Chen et al., 2019). Inputs X to convolutional layers are separated into $X = \{X^{\mathrm{H}}, X^{\mathrm{L}}\}$ along the channel domain. Each layer then computes convolutions on high (X^{H}) and low (X^{L}) frequency parts which are then recombined in the output.

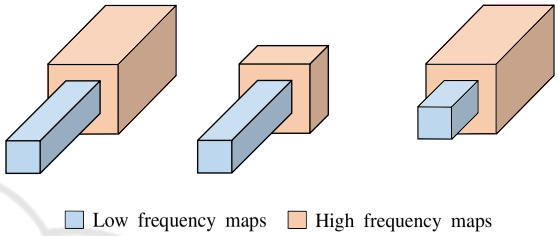


Figure 2: In the original octave convolution, the hyper-parameter α sets the rate of how a fixed number of convolution filters is split into high and low frequency maps. Left to right: $\alpha = 0.5, 0.25, 0.75$. Note: since α changes the network topology, it can not be changed during training.

One of the benefits of the new feature representation is the reduction of the spatial redundancy and the compactness compared with the original representation. Furthermore, octave convolution enable efficient communication between the high and the low frequency component of the feature representation.

3 METHOD

In the following section, we describe our approach which addresses the derivation and integration of the soft octave convolution in GANs.

3.1 Soft Octave Convolution

First experiments with octave convolutions (see Figs. 4, 5 and 6) showed that frequency factorization appears to make GAN training more efficient and stable. However, in the same experiments we also observed that it is quite hard to pick the best value for the hyper-parameter α: While shifting towards more lower frequent convolutions, the GAN training becomes more stable, while at the same time the lack of high frequencies makes the generated images blurry.

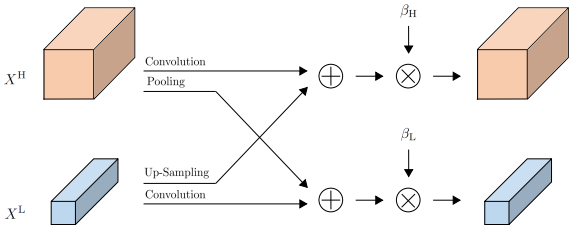


Figure 3: Schematic overview of the soft octave convolution. Inputs X to convolutional layers are separated into $X = \{X^{\mathrm{H}}, X^{\mathrm{L}}\}$ along the channel domain as in the original formulation. We fix $\alpha = 0.5$ and introduce ratio factors $\beta_{\mathrm{L}} X^{\mathrm{L}}$ and $\beta_{\mathrm{H}} X^{\mathrm{H}}$.

To overcome this trade off, we suggest a novel reformulation of the octave convolutions which allows to change the ratio of the frequency factorization during training: the soft octave convolution. Instead of using a fixed factorization ration α, we introduce two independent[1] ratio factors

$$\beta_{\mathrm{L}} X^{\mathrm{L}} \quad \text{and} \quad \beta_{\mathrm{H}} X^{\mathrm{H}}. \tag{3}$$

Fig. 3 shows a schematic overview of our soft octave convolutions. Setting a fixed $\alpha = 0.5$, we use the β weights to shift the ration between high and low frequencies. This allows us to to apply different training schedules, e.g. forcing the GAN to learn the low frequency parts of an image before the high frequent details (see Fig. 9 for example ratio schedules). The results in the experimental section show, that this way, GANs with soft octave convolutions are even more stable and produce high detail images.

3.2 Model Architecture

We use common GAN architectures and simply replace all standard convolutions in the generator as well as in the discriminator with the original octave (Chen et al., 2019) and our proposed soft octave convolution. Such a change has almost no consequences on the architecture elements since it has been designed in a generic way making it a plug-and-play component. However, octave convolution has some impact on batch normalization layers. This regularization technique expects to have as input the same amount of activations from the feature maps. Because of the octave convolution nature, the size of feature maps will diverge between low and high frequency maps. To cope with this issue, two independent batch normalizations will be deployed, one for the low and one for the high frequency feature maps.

[1]In special cases β_L and β_H can be coupled like $\beta_L = 1 - \beta_H$.

3.3 Insights on the Frequency Domain

So far, we motivated the use of (soft) octave convolutions by the intuition, that the training process of GANs will become more stable if we force the networks to focus on the coarse (low frequency) image structures in early stages of the training, before adding (high frequent) image details later on. While our experiments confirm this assumption (see Fig. 7), we follow the recent findings in (Durall et al., 2020) for the theoretical analysis of this effect. It has been shown in (Durall et al., 2020), that most up-sampling and up-convolution units, which are commonly used in GAN generators, are violating the sampling theorem. Hence, convolutional filters in the generator are prune to produce massive high frequency artifacts in the output images which can be detected by the discriminator and thus resulting in unstable training runs. While (Durall et al., 2020) propose to use larger convolutional filters and an extra regularization in order to fix this problem, soft octave convolutions allow us to regulate the spectrum of the output images directly. Fig. 10 shows the frequency spectrum of images generated by GANs with soft octave convolutions in comparison with the vanilla case.

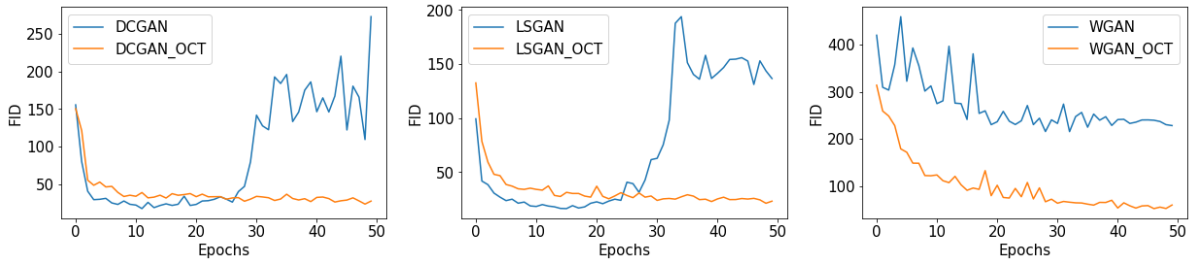
4 EXPERIMENTS

In this section, we present results for a series of experiments evaluating the effectiveness and efficiency of proposed soft octave convolutions. We first give a detailed introduction of the experimental setup. Then, we discuss the results on several different GAN architectures, and finally we explore different configurations modifying the weight of low and high frequency feature maps accordingly. Source code is available on *Github*[2].

4.1 Experimental Settings

We train all architectures on the CelebFaces Attributes (CelebA) dataset (Liu et al., 2015). It consists of 202,599 celebrity face images with variations in facial attributes. In training, we crop and resize the initially 178x218 pixel image to 128x128 pixels. All experiments presented in this paper have been conducted on a single NVIDIA GeForce GTX 1080 GPU, without applying any post-processing. Our evaluation metric is Fréchet Inception Distance (FID) (Heusel et al., 2017), which uses the Inception-v3 network

[2]Source code: https://github.com/cc-hpc-itwm/Stabilizing-GANs-with-Octave-Convolutions



(a) DCGAN baseline. (b) LSGAN baseline. (c) WGAN baseline.

Figure 4: Each figure shows the FID evolution during the training using a certain GAN implementation and its standard octave convolution variant with $\alpha = 0.5$.

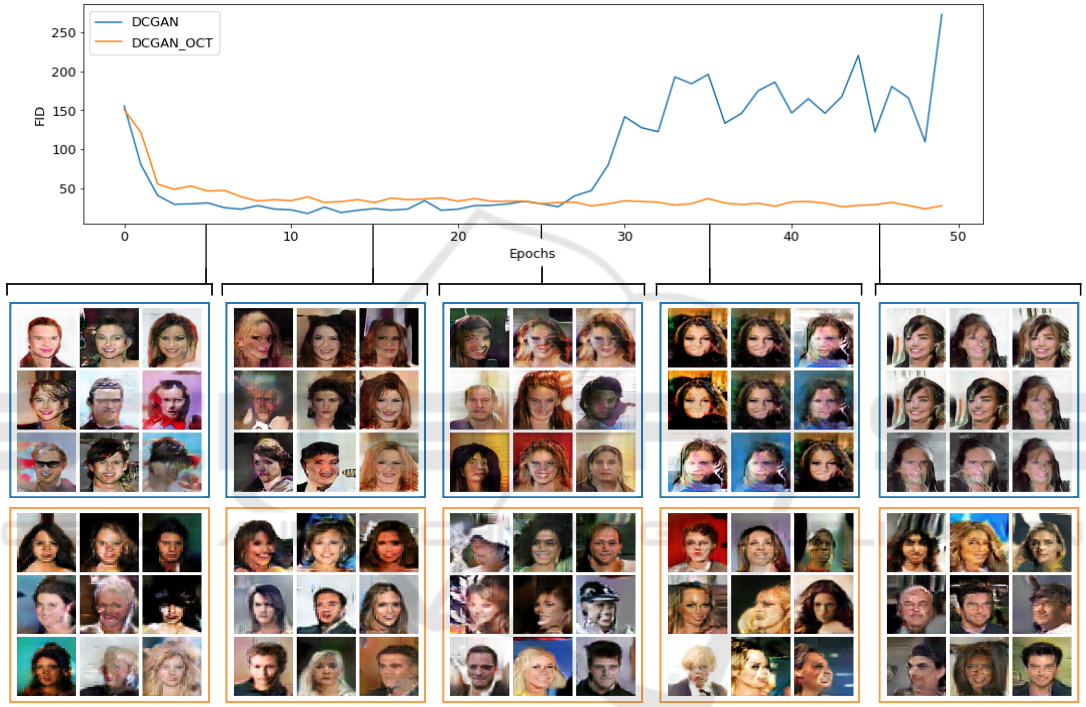


Figure 5: The figure shows the FID evolution together with some random generated examples using a standard DCGAN and its original octave ($\alpha = 0.5$) implementation (DCGAN_OCT).

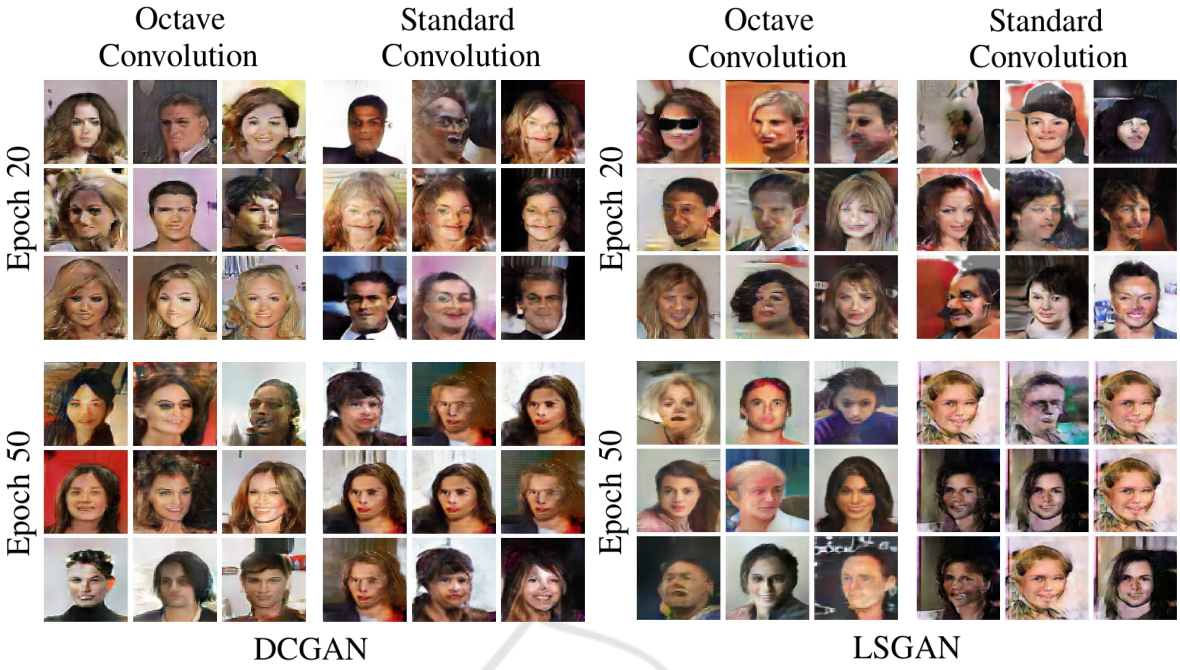
pre-trained on ImageNet to extract features from an intermediate layer. Then, we model the distribution of these features using a multivariate Gaussian distribution with mean μ and covariance Σ. This procedure is conducted for both real images x and generated images z, and it can be written as

$$\text{FID}(x,z) = ||\mu_x - \mu_z||_2^2 + Tr(\Sigma_x + \Sigma_z - 2(\Sigma_x\Sigma_z)^{\frac{1}{2}}). \quad (4)$$

Lower FID is better, corresponding to more similar real and generated samples as measured by the distance between their feature distributions.

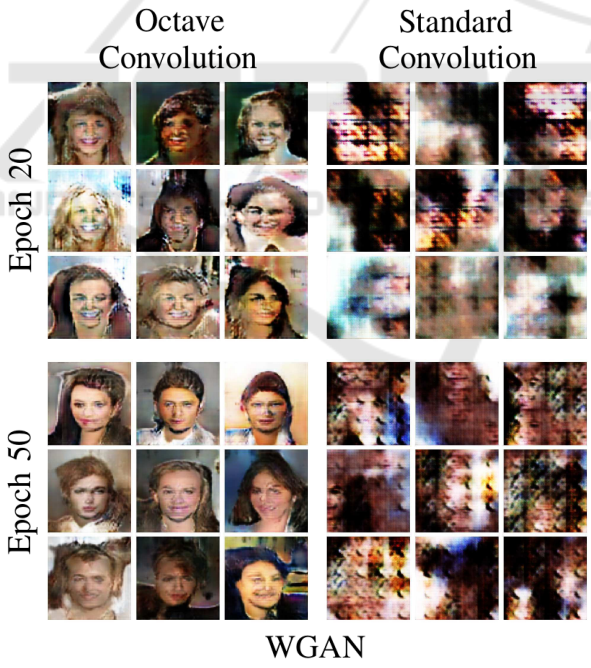
4.2 Training

In this subsection, we investigate the impact of replacing the standard convolution with octave convolution. We conduct a series of studies using well-known GAN baselines which we have not optimized towards the dataset since the main objective here is to verify the impact of the new convolutional scheme and not to defeat state-of-the-art score results. In particular, we constrain our experiments to three types of GANs: DCGAN, LSGAN and WGAN. All comparisons between the baseline methods and the proposals have the same training and testing setting. We use an Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.5$, $\beta_2 = 0.999$ during training in all the cases.



(a) DCGAN baseline.

(b) LSGAN baseline.

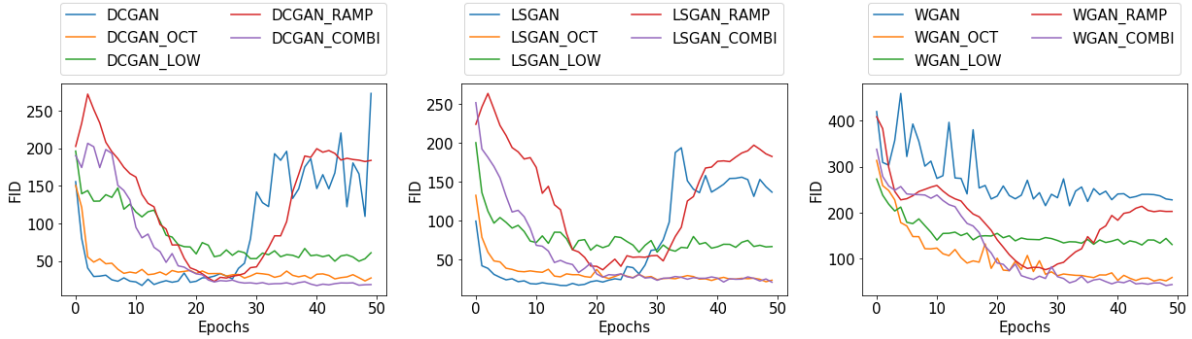


(c) WGAN baseline.

Figure 6: The figures show three independent baselines used in the experiments. Each case contains several samples across two dimensions components: the horizontal and the vertical. The first refers to the type of convolution implemented (original octave or vanilla convolution), and the second represents the stage of the training (epoch) in the particular baseline.

We set the batch size to 64 and run the experiments for 50 epochs. We update the generator after every discriminator update, and the learning rate used in the implementation is 0.0002.

Standard Octave Convolution. First, we conduct a set of experiments to validate the effect of the origi-



(a) DCGAN baseline. (b) LSGAN baseline. (c) WGAN baseline.

Figure 7: FID evolution during the training using different GAN implementation and their octave variants. The suffixes stand for the following: OCT octave convolution with $\alpha = 0.5$ (vanilla configuration), LOW octave convolution with $\alpha = 0.99$, RAMP soft octave convolution with $\alpha = 0.5$ and βs as in 9a, and COMBI soft octave convolution with $\alpha = 0.5$ and βs as in 9b.

nal octave convolution. Therefore, we set the α to 0.5. We begin with using the baseline models and compute the FID after each iteration. Then, we repeat the same procedure but this time we train using the octave convolution on the models. Our results in Fig. 4 show that in all three baselines, the octave model generates images of better or similar quality compared to the previous training. Moreover, we can observe the improvement of stability during training for the octave implementation.

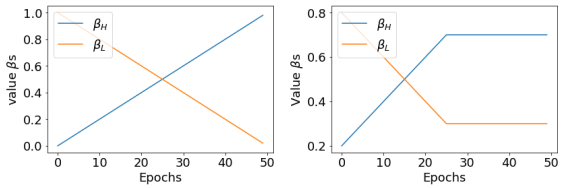
Fig. 5 depicts again the comparison between the vanilla DCGAN with the octave version. However, this time the plot includes an arbitrary set of samples which clearly show that these curves correlate well with the visual quality of the generated samples. Even more detailed and extended qualitative evaluations are presented in Fig. 6, where numerous samples from all the baselines are displayed. Note that vanilla DCGAN and LSGAN start to suffer from mode collapse from epoch 25 forward. Thus, we choose epoch 20 to do a fair qualitative comparison as it seems to be the optimal training epoch. We also show the final results (epoch 50), which support the stability claim held in this work.

Soft Octave Convolutions. In this second part of the experiments, we conduct an analysis of the impact of the low and high frequency feature maps. In order to verify how sensitive GANs are to these modifications, we start running a test for the three baselines, where we set α to 0.99[3] (see Fig. 8). By doing so, we get rid of all the high frequency maps, and as it is expected, the training shows constant stability since low frequencies do not contain big jumps or variations. On the other hand, surprisingly the score results are not dramatically worse than vanilla baselines (see Fig 7). Indeed, it is interesting to notice that both share a similar FID score evolution.

Figure 8: The figure compares two set of random generated images using DCGAN with octave convolution but different αs. (Top) Implemented with $\alpha = 0.5$. (Bottom) Implemented with $\alpha = 0.99$. The bigger the α is, the less amount of high frequency components are present.

From the previous results, we notice the importance of hyper-parameter α. However, it is a well-known NP-hard problem to find the best topology in deep neural networks and in fact, it is an area of active research by itself (Elsken et al., 2018; Liu et al., 2018; Xie et al., 2019). As a consequence, we avoid to modify directly the topology by changing α. Driven by these observations, finally, we conduct a new series of experiments based on two new hyper-parameters β_L and β_H. Indeed, they can be seen as an extension of α because they will modify the feature maps too. Nonetheless, βs do not modify the amount of feature maps, but their weight. In Fig. 9 are plotted two different strategies followed in the work. We first implement a ramp scheme (see Fig. 9a). The intuition behind is that low frequency signals that cap-

[3]We cannot set α to 1 because of implementation issues. Nevertheless, the difference should be negligible.



(a) Ramp. (b) Combination.

Figure 9: These plots show the weight βs curve used in ramp (a) and in combination (b) scheme.

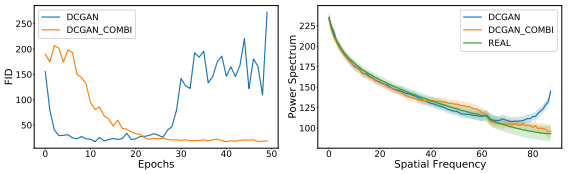
ture the global layout and coarse structure are learnt at the beginning, and after a certain time the high frequency parts that capture fine details, start to appear and gain more importance. Trying to capture such a behaviour, we deploy the ramp evolution. Nonetheless, this strategy might be too harsh as the role played by the low and high frequencies is too insignificant at certain training stages (see Fig. 7). As a result, we introduce a second weighting strategy called combination (see Fig. 9b), which tries to be a trade-off between frequency components offering an optimized combination. In Fig. 7 are shown the three baselines and their octave variants.

4.3 Stability and Effects in the Frequency Domain

Finally, in this subsection we analyse the impact of the soft octave convolution in the frequency domain. In Fig. 10, we compare the standard convolution with the COMBI soft octave convolution. On the one hand, we have the FID curves that describe the stability during training. As we have seen in previous sections, our approach guarantees a much more stable behaviour without having any break. On the other hand, inspired by (Durall et al., 2019; Durall et al., 2020), we take the outputs from the different methods and we compute their spectral components after the training is over. This experiment allows to confirm the effect that our method has on the spectrum domain, being able to correct the artifacts that standards convolutions have in the high frequency band. In other words, soft octave convolution pushes the frequency components towards the real one.

5 CONCLUSIONS

In this work, we tackle the common problem of GAN training stability. We propose a novel yet simple convoluitonal layer coined as soft octave convolution. Intuitively, our approach forces GANs to learn low frequency coarse image structures before descending



(a) FID evolution during the training. (b) 1d power spectrum from the outputs.

Figure 10: These plots depict the comparison between standard convolution and soft octave convolution in terms of stability and frequency domain.

into fine (high frequency) details. As a result, we achieve both a stable training and a reduction of common artifacts present in the high frequency domain of generated images. Furthermore, we show how this method is orthogonal and complementary to existing methods and leads to generate images of better or equal quality suppressing the mode collapse problem. We believe the line of this work opens interesting avenues for feature research, including exploring Bayesian optimizations.

REFERENCES

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

Campbell, F. W. and Robson, J. (1968). Application of fourier analysis to the visibility of gratings. *The Journal of physiology*, 197(3):551–566.

Chen, Y., Fang, H., Xu, B., Yan, Z., Kalantidis, Y., Rohrbach, M., Yan, S., and Feng, J. (2019). Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. *arXiv preprint arXiv:1904.05049*.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258.

DeValois, R. L. and DeValois, K. K. (1990). *Spatial vision*. Oxford university press.

Durall, R., Keuper, M., and Keuper, J. (2020). Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7890–7899.

Durall, R., Keuper, M., Pfreundt, F.-J., and Keuper, J. (2019). Unmasking deepfakes with simple features. *arXiv preprint arXiv:1911.00686*.

Elsken, T., Metzen, J. H., and Hutter, F. (2018). Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777.

Han, S., Pool, J., Narang, S., Mao, H., Gong, E., Tang, S., Elsen, E., Vajda, P., Paluri, M., Tran, J., et al. (2016). Dsd: Dense-sparse-dense training for deep neural networks. *arXiv preprint arXiv:1607.04381*.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

Ke, T.-W., Maire, M., and Yu, S. X. (2017). Multigrid neural architectures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6665–6673.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Lin, C. H., Chang, C.-C., Chen, Y.-S., Juan, D.-C., Wei, W., and Chen, H.-T. (2019). Coco-gan: Generation by parts via conditional coordinating. *arXiv preprint arXiv:1904.00284*.

Liu, H., Simonyan, K., and Yang, Y. (2018). Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738.

Luo, J.-H., Wu, J., and Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066.

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. (2017). Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Oord, A. v. d., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Xie, S., Kirillov, A., Girshick, R., and He, K. (2019). Exploring randomly wired neural networks for image recognition. *arXiv preprint arXiv:1904.01569*.