How to Exploit the Transferability of Learned Image Compression to Conventional Codecs

Jan P. Klopp  
National Taiwan University  
kloppjp@gmail.com

Keng-Chi Liu  
Taiwan AI Labs  
calvin89029@gmail.com

Liang-Gee Chen  
Shao-Yi Chien  
National Taiwan University  
lgchen@ntu.edu.tw  sychien@ntu.edu.tw

Abstract

Lossy image compression is often limited by the simplicity of the chosen loss measure. Recent research suggests that generative adversarial networks have the ability to overcome this limitation and serve as a multi-modal loss, especially for textures. Together with learned image compression, these two techniques can be used to great effect when relaxing the commonly employed tight measures of distortion. However, convolutional neural network-based algorithms have a large computational footprint. Ideally, an existing conventional codec should stay in place, ensuring faster adoption and adherence to a balanced computational envelope.

As a possible avenue to this goal, we propose and investigate how learned image coding can be used as a surrogate to optimise an image for encoding. A learned filter alters the image to optimise a different performance measure or a particular task. Extending this idea with a generative adversarial network, we show how entire textures are replaced by ones that are less costly to encode but preserve a sense of detail.

Our approach can remodel a conventional codec to adjust for the MS-SSIM distortion with over 20% rate improvement without any decoding overhead. On task-aware image compression, we perform favourably against a similar but codec-specific approach.

1. Introduction

Forming the basis for video compression, image compression’s efficiency is vital to many data transmission and storage applications. Most compression algorithms are lossy, i.e., they do not reproduce the original content exactly but allow for deviations that reduce the coding rate.

Lossy compression optimises the objective

\[ \mathcal{L} = R + \lambda D \]  

where \( R \) and \( D \) stand for rate and distortion, respectively, and \( \lambda \) controls their weight relative to each other. In practice, computational efficiency is another constraint as at least the decoder needs to process high resolutions in real-time under a limited power envelope, typically necessitating dedicated hardware implementations. Requirements for the encoder are more relaxed, often allowing even offline encoding without demanding real-time capability.

Recent research has developed along two lines: evolution of exiting coding technologies, such as H264 [41] or H265 [35], culminating in the most recent AV1 codec, on the one hand. On the other hand, inspired by the success of deep learning in computer vision, based on the variational autoencoder [20], several learned image compression approaches have been developed. Through careful modelling of the latent code’s symbol probabilities, those learned codecs have recently shown to outperform the x265 codec on various performance measures [30]. This trades engineering effort for complexity: training a neural network image codec is simple compared to designing a conventional codec with all its bells and whistles. However, the resulting neural decoder requires on the order of \( 10^5 \) operations per pixel, where a conventional codec has at most a few hundred operations, being several orders of magnitude more efficient. This is an important factor, given how much video and imagery is consumed on mobile devices.

Another advantage of learned compression is the ability to be easily adapted to different distributions or distortion measures. By merely collecting data and choosing an appropriate distortion loss, a new codec can quickly be trained without having to manually re-engineer coding tools or internal parameters.
In this work, we take a step towards leveraging the adaptability of learned codecs for conventional codecs without changing the conventional codec itself. In doing so, we enable a higher compression performance without touching the decoder, meaning that existing endpoint decoder implementations do not require changes.

In lossy compression, adaptability has two aspects: dropping information that is hard to encode and of little importance and, secondly, exploiting expensive redundancies in the data. The latter would require either reorganising data or changing the coding tools a codec can apply at encoding time. The former, though, is feasible as the data can be changed prior to letting the encoder map it to its coded representation.

We demonstrate three possible applications of this scheme:

1. We apply a filtering mechanism to siphon out information that would drive the coding cost but is only of little concern with respect to the chosen loss function. Introducing a learned codec as a surrogate gradient provider conventional codecs can be refitted to the MS-SSIM distortion measure at over 20% rate improvement.

2. We apply the approach to task-specific compression where we perform favourably on image classification without having to model the target codec specifically.

3. We pair the surrogate-induced filter with a generative adversarial network to alter an image beyond optimisation for simple distortion measures. Textures are replaced with perceptually similar alternatives that have a shorter code representation, thereby survive the coding process and preserve the crispness of the original impression.

By simply manipulating the input data, we alter the way the fixed-function codec behaves. We present evaluations across different datasets and codecs and show examples of how introducing GANs can lead to distinctly sharper-looking images at low coding rates.

### 2. Learning to Filter with a Surrogate Codec

Current coding mechanisms are most often instances of transform coding, where the data is first transformed, then quantised in the transform domain and finally encoded losslessly using a variant of arithmetic coding. Conventional codecs use a cosine- or sine-transform, while learned codecs replace them with a convolutional neural network. This has the distinct advantage that the filters of the neural network can be adapted to leave out information that has low marginal coding gain, i.e. costly to encode and with only little benefit as measured by a given loss function. Table 1 shows which functionality is invoked by different coding tools.

In codecs with fixed transforms, this process needs to be emulated manually, which typically happens by varying the quantisation granularity so that high-frequency transform coefficients are quantised more coarsely than low-frequency ones. Different rate-distortion trade-offs are then achieved by adjusting the granularity for all coefficients simultaneously. Omitting information happens implicitly through the quantisation process. This, however, requires the transforms to separate information according to its importance so that varying quantisation granularity has the desired effect. The general underlying idea is that low-frequency content is the most important and importance decreases towards higher frequencies. While this adequately distinguishes noise from actual information for some content, it does not hold in general. Hence, more adaptive filters are desirable, however difficult to adjust as their influence on the coding behaviour is unknown.

Omitting unimportant information helps to significantly reduce the size of compressed multimedia data. For an optimal codec, it would be desirable to design the data filtering process with the distortion measure on the one hand and the transformation and coding procedure on the other in mind. Machine-learning based codecs can do this implicitly because all elements required are differentiable. Their optimisation procedure allows the filters to adapt to the input data, the distortion measure, and the arithmetic coding procedure. For conventional codecs, this process is realised through extensive engineering and by relying on local optimisation of the code structure, for example, how to partition the image into transform blocks or the choice of prediction modes, which makes it difficult to turn a codec into a differentiable function and obtain a gradient with respect to the input image.

In the machine learning literature this has been addressed as black-box function optimisation, for example through the REINFORCE learning algorithm [42] or the learning-to-learn approaches [15, 3, 11]. These methods rely on an approximation of the unknown function to be optimised, though. Filtering of less important information is likely

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv. Codec</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Learn. Codec</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Denoising</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Inpainting</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>[12, 34]</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Our’s / [37]</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
to require such an approximation to be very precise, as the changes applied to the input image are expected to be small in magnitude. In preliminary experiments, we found that even a complex, deep model does not approximate a codec’s rate and its decoded image accurately. This is most likely due to the complexity of the non-stationary error distribution that codecs produce, which varies between adjacent blocks.

To address this, [12] present an approach for predicting an optimised quantisation matrix for the JPEG codec. [34] adds a pre-editing network to this, similar to our approach. [37] also proposes image editing prior to encoding for better coding performance. All three approaches demonstrate effectiveness, but lean on modelling the JPEG codec as a differentiable function. Hence, they are limited to this rather simple codec. In the case of [12], their exclusive use of quantisation tables imposes restrictions on the corrections they can perform.

In contrast, we present an approach that does not rely on approximating the codec directly but exploits a (differentiable) model that optimises the same objective, i.e. Eq. 1. The structure of this approach is depicted in Figure 1. Our objective is to approximate the gradient

$$
\frac{\partial L_h}{\partial I} = \frac{\partial h_R(I)}{\partial I} + \lambda \frac{\partial D(h_D(I), I)}{\partial I}
$$

(2)

of the loss function $L_h$ of a codec $h$ with a rate $h_R(I)$, a decoded reconstruction $\hat{I} = h_D(I)$, and a distortion measure $D(\hat{I}, I)$ between the decoded and the original image $I$. Image compression models like [30] provide a differentiable codec $g$ with rate $g_R(I; \theta_c)$ and decoded reconstruction $g_D(I; \theta_c)$ both utilising some parameter vector $\theta_c$. Using this, an image $I$ can be manipulated by adding $\zeta^*$ where

$$
\zeta^* = \arg \min_\zeta L_g = \arg \min_\zeta g_R(I + \zeta) + \lambda_T D_T (g_D(I + \zeta), I)
$$

(3)

can be optimised using a gradient descent based algorithm. The original codec then operates on the modified input image $\hat{I} = I + \zeta^*$ with the objective of achieving

$$
L_h > \hat{L}_h = h_R(\hat{I}) + \lambda_T D_T (h_D(\hat{I}), x).
$$

(4)

for a chosen target distortion $D_T$. The assumption here is that, though the losses differ ($L_g \neq L_h$), the provided gradients are similar:

$$
\frac{\partial L_g}{\partial I} \approx \frac{\partial L_h}{\partial I}.
$$

(5)

We introduce a filter $\hat{I} = f_F(I; \theta_F)$ to predict the filtered image $\hat{I}$ directly from the original image $I$. Its parameters $\theta_F$ can be trained using the gradient of the loss $\hat{L}_g$ where the surrogate codec provides the rate estimation. The parameter $\lambda_T$ controls the trade-off between rate and distortion for the filter $f_F$.

Most codecs are designed for a particular intrinsic distortion measure $D_I$. For common codecs like H265 or H264, coding tools and mechanisms were evaluated using PSNR during standardisation. This suggests that a codec may perform differently in terms of the rate-distortion trade-off.

Figure 1. Structural overview of our method. The goal is to obtain a trained filter $f_F(I; \theta_F)$ to optimise the input image $I$ for encoding by a target codec. This target is typically not differentiable. A surrogate codec is used instead. It provides a differentiable rate estimate. For the reconstruction there are two options as indicated by the switch. The first is to take the surrogate’s reconstruction, the second to invoke the target during the forward pass. The gradient flows back accordingly either through the surrogate’s reconstruction or directly into the filter. The target distortion measure $D_T$ can be chosen freely. The surrogate codec is pre-trained with a distortion measure similar to the one of the target codec so as to imitate its behaviour. At testing time, the filter is applied to the image before it’s encoded by the target codec.
when subjected to a different distortion measure. It is therefore likely that the measure used to optimise the parameters \( \theta_C \) of the learned codec \( g \) should be the same as that for which \( h \) was designed. This will be confirmed by experiments. We will denote that measure by \( D_S \) as it is the one to be approximated. When optimising Equation 3 on the other hand, the distortion \( D_T \) can be freely chosen so long as it is differentiable.

With this framework in place, we are set to optimise the filter \( f_F \) for a chosen particular target distortion \( D_T \) while observing rate constraints as given by the surrogate codec’s rate estimation.

### 2.1. Network Architectures

#### 2.1.1 Filter

The filter needs to remove information that is too costly to encode and has little influence on the distortion \( D_T \). The chosen network architecture is depicted in Fig. 2. The first stage is identical to the encoder and the rate estimation of the chosen surrogate codec. The general idea is that the filter first needs to estimate which regions of the image have a higher concentration of information. This is contained in the entropy estimates \( H_{i,j,k} \). \( H_{i,j,k} \) is of dimension \( \frac{b}{16} \times \frac{w}{16} \times C \) where \( C \) is the number of channels of the code tensor and \( b \) and \( w \) the dimensions of the image. For the surrogate model from [30] chosen here, this is \( C = 320 \). The next step is to successively scale the code up to the size of the image using four layers of transposed convolutions with a stride of two. We keep reducing the number of filters in each layer so as to have 16 output features per pixel. The idea behind not using a single value at each pixel is that we would like to represent a more complex distribution as the allocation of coding cost to single pixels is obviously challenging and a distribution can be interpreted as being conditional on the input image that is fused in the next step. The resulting pixel-wise entropy estimate \( H_{\text{pixel}} \in \mathbb{R}^{h \times w \times 16} \) is concatenated with the input image \( I \) to serve as input to the actual filter. The filter itself consists of four residual blocks followed by a convolutional layer at the end. We use 64 channels and kernels of size \( 3 \times 3 \) throughout the entire filter.

#### 2.1.2 Discriminator

We choose a simple discriminator, as its only task is to recognise artefacts in the filtered image \( \hat{I} \) and simple patterns in texture as opposed to larger structures or semantically coherent objects when images are generated from scratch. Our discriminator has three consecutive stages to capture artefacts at different scales, as previously described by [32]. Each stage consists of two residual blocks. Subsequent stages apply \( 2 \times 2 \) average pooling prior to the residual blocks. Each stage \( s \) is followed by a single convolutional filter that computes the discriminator’s prediction \( p_s \). To fuse those predictions, we first apply a sigmoid to convert them into probabilities before computing the mean over each stage:

\[
\hat{y}_{GAN}(I) = \frac{1}{3} \sum_{s=1}^{3} E \left[ \text{sigmoid} \left( p_s(I) \right) \right]
\]

We noticed that only computing high-level outputs at the discriminator does lead to low-level artefacts like blurriness in the image, an observation shared previously by Rippel and Bourdev [32]. Similar to [2], we used a least-squares objective to train the discriminator, as previously introduced in [27]. Discriminator and filter are optimised alternatingly, i.e. not using the same batch, for stability.

### 3. Experiments

#### 3.1. Experimental Conditions

##### 3.1.1 Datasets

used for evaluation are the Kodak dataset (24 images, about 0.35MP) and the “professional” validation dataset of the CLIC competition (41 images at more than 2MP). Our training data is sampled from the ImageNet dataset and CLIC’s provided training data. We randomly resize an image so that its shorter side is between 512 and 1024px and then sample a \( 256 \times 256 \) crop.

##### 3.1.2 Surrogate Model

We use the deep convolutional neural network model proposed by Minnen et al. [30]. Besides a 4-layer encoder and decoder with generalised divisive normalisation [7], it uses context modelling to minimise the entropy of the code symbols. It has been shown to work well for both the MS-SSIM as well as the PSNR distortion metrics and achieves state-of-the-art performance on the Kodak test set. In addition, there is a reference implementation available [5, 9].

##### 3.1.3 Optimization

Our models are implemented using PyTorch [31] and trained with the Adam [19] optimizer. A batch size of 8 is used with a learning rate of 1e-4 for 400.000 iterations, which is reduced to 1e-5 for another 100.000 iterations afterwards.

##### 3.1.4 Codecs

We use JPEG, WebP, BPG (H265/HEVC) [10] and the recently released AV1 to validate our hypothesis and demonstrate coding improvements. For all codecs, we choose maximum gain over encoding speed and encode in the codec’s native YCbCr colour space.
3.2. Adaptation to MS-SSIM

We train a filter to minimise the rate and the impact on the MS-SSIM distortion measure. The idea is that most codecs were designed to minimise the MSE and also optimise that measure internally when choosing the best representation at encoding time. Not only can the MSE be computed faster, but it also doesn’t transcend block boundaries, making it a computationally suitable objective. Hence, “re-targeting” a codec to MS-SSIM without having to change its encoder is appealing if it can be done by simply adding a filter at encoding time.

We train three models at different rate-distortion trade-offs \( \lambda_T \) to cover different parts of the rate range. We found in preliminary experiments that, when optimising \( \lambda \) directly, i.e. not training a filter \( f_T(\lambda; \theta_T) \), learned surrogate codecs perform well around their operating point as determined by \( \lambda_S \), however, quickly diminish in performance once the trade-off point \( \lambda \) is changed significantly. For this set of experiments, we chose \( \lambda_S \in \{0.1, 0.2, 0.4\} \) for the MSE-optimized surrogate. When optimising Eq. 4 using the surrogate codec, the rate-distortion trade-off \( \lambda_T \) for the target MS-SSIM loss needs to be chosen appropriately. We found \( \lambda_T = 500\lambda_S \) to work well by comparing at which pairs of \( \lambda_S, \lambda_T \) surrogate codecs optimised for each of the two distortions would yield similar rates.

There are limitations for very high bit rates as our re-targeting process takes out information, and hence the reconstruction error cannot be arbitrarily small. This is reflected in the graphs in Fig. 3. Towards higher rates, where the error is very small, the filtering effect fades, with the filtered version eventually having a worse distortion than the original one. For the more advanced codecs AV1, BPG and WebP, the rate savings peak for low rates. Only for JPEG, the rate savings are much less dynamic. The reason may lie in the simplicity of the JPEG codec, with little space for optimisation as the coding and quantisation process is static, only the quantisation tables can be optimised (over the entire image). Interestingly, for both datasets, the BPG codec initially performs worse than AV1. However, it has higher adaptation coding gains so that the filtered variants of both codecs are almost on par.

Comparing the rate savings of the Kodak dataset in Fig. 3 to those of the validation part of the CLIC Professional dataset in Fig. 4, one notices that the three advanced codecs...
show similar behaviour in terms of rate savings. There’s a peak early on, followed by a steady, parallel decline. This indicates that, at higher rates, the induced error from the filtering process is too high relative to the coding error. When optimising only for numeric measures instead of perceptual ones, this limitation cannot be overcome. This is another reason to also look at perceptual losses like generative adversarial networks, as discussed in Section 3.3. Beyond this, the higher resolution images of the CLIC Professional dataset are more difficult to optimise yet still have coding gains up to 15%. Higher resolutions often have a lower information density, making it harder to filter out irrelevant parts.

Lastly, we also test with a surrogate codec optimised for $D_S = \text{MS-SSIM}$. This is shown as a dotted cyan coloured line in Fig. 3. There is no improvement over the raw codec, showing that the surrogate’s rate estimation needs to be based on the correct loss function to model a codec’s rate allocation properly.

3.3. Generative Adversarial Network

Generative Adversarial Networks have led to remarkable results in various tasks where high-resolution natural images are to be reconstructed, such as super-resolution [23]. In the context of codec adaptation, the GAN serves as an (additional) loss measure.

Following [2], we employ MSE $\mathcal{L}_{\text{MSE}}$ and the VGG-based perceptual loss $\mathcal{L}_{\text{VGG}}$ proposed in [30] in addition to the GAN loss $\mathcal{L}_{\text{GAN}}$ to ensure that the image content remains similar and only textural information changes. The final loss function for the filter adds these three distortion terms to the rate term $h_R$ of the surrogate codec:

$$\mathcal{L}_{\text{Filter}} = \gamma_{\text{GAN}} \mathcal{L}_{\text{GAN}} + \gamma_{\text{VGG}} \mathcal{L}_{\text{VGG}} + \gamma_{\text{MSE}} \mathcal{L}_{\text{MSE}} + h_R \quad (7)$$

where the different loss components can be weighted according to one’s objectives. The filter’s GAN loss $\mathcal{L}_{\text{GAN}}(\hat{I})$ conditioned on the filtered image $\hat{I} = f_F(I; \theta_F)$ is given by the least squares objective from [27]:

$$\mathcal{L}_{\text{GAN}}(I) = E_I \left[ (1 - y_{\text{GAN}}(f_F(I; \theta_F)))^2 \right] \quad (8)$$

where $y_{\text{GAN}}(\hat{I})$ is defined by Eq. 6. The discriminator’s loss optimizes the opposite objective of $\mathcal{L}_{\text{GAN}}$:

$$\mathcal{L}_{\text{Discriminator}} = \mathbb{E}_I \left[ (1 - y_{\text{GAN}}(I))^2 \right] + \mathbb{E}_I \left[ y_{\text{GAN}}(f_F(I; \theta_F))^2 \right] \quad (9)$$

$\mathcal{L}_{\text{Discriminator}}$ and $\mathcal{L}_{\text{Filter}}$ are optimized in turn every other iteration.

The choice of weights in Eq. 7 determines what kind of sensitivity the GAN discriminator should apply. A lower weight $\gamma_{\text{GAN}}$ allows only for small changes in the image so that smaller artefacts are being corrected. However, textures will not be exchanged for ones that have a shorter code length yet look crisper when coded at low rates.

Setting the GAN’s weight to $\gamma_{\text{GAN}} = 5.0$ and $\gamma_{\text{VGG}} = 0.01, \gamma_{\text{MSE}} = 0.001$, the discriminator forces the filter to replace textures that are indistinguishably sharp yet have a shorter code length. The MSE and the VGG-based loss ensure that the texture is similar to the original. Examples of such a transformation are shown in Table 2. Due to the low rate, the codec’s aggressive quantisation causes most details to wash out. At even lower coding rates, our pre-filtered images are able to preserve perceived details throughout the coding process.

This way, without changing the mechanisms a conventional codec employs to represent an image, we can manipulate the codec through its input into optimising for a rather perception-based objective. The benefit is that the shallow decoder that operates with only about 100 operations per pixel can be reused, and the encoder is simply augmented and otherwise remains unchanged.

3.4. Task-Aware Compression

Task-aware compression takes a non-perceptual task into account when compressing data. In this work we follow [12] and use the ImageNet [17] object recognition task. To learn the filter, we use the pre-trained ResNet-18 [14] available from PyTorch. For this task, the filter receives the classifier’s feedback on which features are important. This necessitates two changes to our algorithm.

The first is to initialise the filter with a very low variance so that the generated correction signal has a very low amplitude at the beginning of the training. If one omits this, the (initially random) correction signal will still be adjusted to strengthen certain features. However, the learned codec’s rate estimate doesn’t transfer well. Apparently, the reason for this is that there are certain high-frequency signals that are cheap for a learned but expensive for a conventional codec and the filter tends to exploit those because they are closer to its own starting point, i.e. the high variance random initialisation.

The second measure is to use the target codec’s decoded version of the filtered image $\hat{I}, h_D(\hat{I})$, to obtain feedback from the classifier about which features are important.
Table 2. Visual comparison, showing how our algorithm modifies certain textures within the image to mimic the original image’s details with lower-code-length content as controlled by a generative adversarial network.

<table>
<thead>
<tr>
<th>Original</th>
<th>H265 (unfiltered)</th>
<th>Ours (H265 filtered)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image 1" /></td>
<td><img src="image2.png" alt="Image 2" /></td>
<td><img src="image3.png" alt="Image 3" /></td>
</tr>
<tr>
<td>Rate: 0.086 bit/pixel</td>
<td>Rate: 0.084 bit/pixel</td>
<td></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image 4" /></td>
<td><img src="image5.png" alt="Image 5" /></td>
<td><img src="image6.png" alt="Image 6" /></td>
</tr>
<tr>
<td>Rate: 0.045 bit/pixel</td>
<td>Rate: 0.042 bit/pixel</td>
<td></td>
</tr>
<tr>
<td><img src="image7.png" alt="Image 7" /></td>
<td><img src="image8.png" alt="Image 8" /></td>
<td><img src="image9.png" alt="Image 9" /></td>
</tr>
<tr>
<td>Rate: 0.190 bit/pixel</td>
<td>Rate: 0.192 bit/pixel</td>
<td></td>
</tr>
</tbody>
</table>
While this slows training down due to $B$ calls to the conventional codec during the forward pass for batch size $B$, it resolves the same issue as in the previous paragraph. The information that goes missing is likely to be high-frequency signals involved in rather fine-grained recognition. If the surrogate codec’s decoded image, $g_D(\hat{I})$, was used directly, the filter may overfit and exploit low-cost, high-frequency signals.

For testing, in accordance with [12], we use a pre-trained Inception-v3 architecture [36]. We train a single model using JPEG at a quality level of 30 at training time. At test time, we filter the images once and then encode them at different quality levels. The results are shown in Fig. 5. Although [12] has a more complicated model that mimics the JPEG codec, our approach performs favourably. One reason may be that [12] only manipulates quantisation tables while our approach can apply arbitrary changes to the image. In this, our method, though simpler, is more powerful.

4. Related Work

The works [12, 34, 37] are related to ours in that they improve a conventional codec by altering its input. Their methods, however, are limited to the JPEG codec by explicit modelling, which is difficult to extend to more complex codecs. In the experiments section, we showed that our approach performs similarly to them when retargeting a codec to another metric and performs favourably on task-specific compression.

Our work is related to other approaches that employ generative adversarial networks to reconstruction tasks. For image compression, these are the recent works [2, 29]. Different from our approach, their work treats the problem from a conceptual perspective, ignoring the complexity constraints or the benefit of using existing coding schemes by employing many-layered neural networks in the image formation process.

Similar to their work, we employ techniques from learned image compression. Recursive architectures have been proposed [38, 39, 18] as well as adoptions of variational auto-encoders [7, 24, 4, 1, 32] with various extensions to the latent code model [6, 25, 22, 30, 8, 28]. Within a few years, these techniques have closed the gap to conventional codecs. However, their decoders require three to four orders of magnitude more operations per pixel. Again, this is an important difference to our approach, as we target an existing coding pipeline. Furthermore, those works have previously been extended into the video coding domain, typically by adding and encoding optical flow [26, 13, 33].

Other works have improved upon existing codecs by adding neural networks as denoising modules. Pretrained denoisers such as [16, 43] are effective but, like the aforementioned approaches, do add heavily to the decoder complexity. Using an online adaptation of a small neural network, [21] showed that the complexity could be reduced significantly while delivering a coding gain similar to pretrained networks. On the downside, these methods require changes to the codec architecture and target only numerical losses, i.e. optimising Equation 1, no perceptual ones.

5. Conclusion

We have demonstrated how machine learning-based image compression algorithms can be repurposed as surrogate gradient generators to train image filters to alter the coding characteristic of conventional codecs without explicitly modelling their behaviour. Experiments demonstrate retargeting to the MS-SSIM distortion measure leads to over 20% coding gain, an advantage over a codec-specific optimisation approach on task-aware image compression, and how adding a generative adversarial network can enable the preservation of sharp textures even in very low rate conditions.

Our approach opens up a number of possible directions for future research. Modelling existing or upcoming codecs to a higher accuracy could enable even better results. Traditional, hand-optimised image signal processing pipelines, as they are found in smartphones, for example, could be optimised end-to-end from acquisition to image formation and to coding. The introduced concepts extend to the motion picture domain as well, which could enable codec refitting beyond still images.

Finally, the proposed technique does not require changing the coding pipeline. Encoder and decoder stay in place. Hence, in contrast to many other image coding improvements, it can utilise existing efficient hardware implementations of conventional decoders.
References


