Source Compression with Bounded DNN Perception Loss for IoT Edge Computer Vision

Xiufeng Xie
Hewlett Packard Labs
xiufeng.xie@hpe.com

Kyu-Han Kim
Hewlett Packard Labs
kyu-han.kim@hpe.com

ABSTRACT

IoT and deep learning based computer vision together create an immense market opportunity, but running deep neural networks (DNNs) on resource-constrained IoT devices remains challenging. Offloading DNN inference to an edge server is a promising solution, but limited wireless bandwidth bottlenecks its end-to-end performance and scalability. While IoT devices can adopt source compression to cope with the limited bandwidth, existing compression algorithms (or codecs) are not designed for DNN (but for human eyes), and thus, suffer from either low compression rates or high DNN inference errors. This paper presents GRACE, a DNN-aware compression algorithm that facilitates the edge inference by significantly saving the network bandwidth consumption without disturbing the inference performance. Given a target DNN, GRACE *(i)* analyzes DNN's perception model *w.r.t* both spatial frequencies and colors and *(ii)* generates an optimized compression strategy for the model – one-time offline process. Next, GRACE deploys thus-generated compression strategy at IoT devices (or source) to perform online source compression within the existing codec framework, adding no extra overhead. We prototype GRACE on JPEG (the most popular image codec framework), and our evaluation results show that GRACE indeed achieves the superior compression performance over existing strategies for key DNN applications. For semantic segmentation tasks, GRACE reduces a source size by 23% compared to JPEG with similar interference accuracy (0.38% lower than GRACE). Further, GRACE even achieves 7.5% higher inference accuracy than JPEG with a commonly used quality level of 75 does. For classification tasks, GRACE reduces the bandwidth consumption by 90% over JPEG with the same inference accuracy.

MobiCom '19, October 21–25, 2019, Los Cabos, Mexico

ACM ISBN 978-1-4503-6169-9/19/10…$15.00
https://doi.org/10.1145/3300061.3345448

CCS CONCEPTS

• **Theory of computation → Data compression**; • **Computing methodologies → Image segmentation**; • **Computer systems organization → Neural networks**.

KEYWORDS

mobile edge computing, neural networks, image compression

ACM Reference Format:
Xiufeng Xie and Kyu-Han Kim. 2019. Source Compression with Bounded DNN Perception Loss for IoT Edge Computer Vision. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19), October 21–25, 2019, Los Cabos, Mexico.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3300061.3345448

1 INTRODUCTION

Computer vision (CV) with deep neural networks (DNN) has vast potential in the IoT regime, with promising applications such as object classification for a solar-powered wireless camera and instance segmentation for a city-scale public-safety drone. However, running DNN applications on IoT devices remains fundamentally challenging, due to the limited computing power, storage space, or battery life of the devices. With the upcoming 5G network that features mobile edge computing (MEC), a promising solution to the above challenges is to offload the DNN inference tasks to powerful edge servers close to the clients. Briefly, IoT devices can stream captured video/image source to edge servers, which then perform compute-intensive DNN inference and respond with the results. DNN-based inference at the edge has recently become a new frontier of deep-learning research, drawing enormous attention from IT industry like Hewlett Packard Enterprise [19], Facebook [46], Google [16], and Microsoft [32].

However, with the growing number of wireless IoT devices, limited bandwidth is becoming a fundamental challenge, hindering the deployment of the edge inference for DNN-based applications. DNNs are sensitive to input noise – a single corrupted pixel may fool the DNN [42] – Hence, IoT devices must send high-quality image/video to the edge for inference, causing scalability issues at wireless links. One may

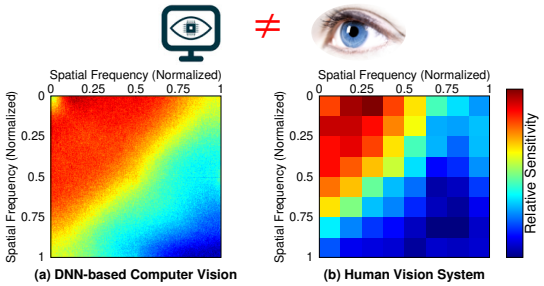


Figure 1: DNN and human eyes have different perceptual sensitivity *w.r.t.* spatial frequencies.

ask if existing codecs would help. Our experiments on DNN-based semantic segmentation (§7.1.1), however, reveal that JPEG compression causes 8% of accuracy drop under typical quality level 75. The lossless PNG produces a more than 2 MB average file size for a high-quality dataset (Cityscapes), which is too large for the edge inference architecture to scale. The poor performance of prevailing codecs when targeting DNNs roots in their human-centric design, *i.e.*, they make compression artifacts invisible specific to human. However, a DNN's perception model is very different from that of human eyes[1], and compression artifacts invisible to human may cause significant DNN accuracy loss, as revealed in [42].

This paper presents a DNN-aware compression algorithm called ***Gra**dient-driven **C**ompression for **E**dge* or GRACE, which saves the network bandwidth without compromising the DNN inference accuracy. GRACE characterizes the perception of a target DNN model based on its gradient *w.r.t.* input components, then customizes the compression strategy accordingly. More specifically, a DNN's gradient of loss *w.r.t.* an input component indicates how much the loss would change when a small noise adds to this input component, and compression artifacts are such noises added to the DNN input. Therefore, GRACE uses a DNN's gradients *w.r.t.* its input to determine a compression strategy that balances the inference accuracy and compression ratio.

The GRACE runs in the following three steps: *(i)* Given a target DNN, GRACE estimates its perception model *w.r.t.* both the spatial frequencies (§4.3) and colors (§4.4), based on the DNN's gradient *w.r.t.* the frequency- & color- components of its input. GRACE uses only a few test images (§6) to probe such a perception model offline on an edge/cloud server. The probing makes no modification to the DNN, as GRACE compresses *for* the DNN, not *by* the DNN. *(ii)* With the estimated perception model, GRACE solves a set of optimization problems (§5.1 and §5.2) to determine the optimal *compression strategy* defined as a set of codec parameters. The optimization also runs offline on an edge/cloud server, and then the optimized compression strategy is distributed to IoT devices. *(iii)* IoT devices run the optimized strategy within the existing codec framework (§2) to compress the video / image inputs, and finally stream the compressed inputs to the edge.

We have prototyped GRACE mainly using PyTorch [34] and performed an extensive evaluation. Our prototype reuses the popular JPEG codec framework (§6), in which the compression strategy is a set of configurable parameters, *i.e.*, the quantization table and the weights of RGB-to-YUV conversion, and thus its complexity and overhead are identical to JPEG. We evaluate GRACE on essential CV tasks, including semantic segmentation and image classification, under various configurations and DNN models (§7). The evaluation results show the promising performance of GRACE. For semantic segmentation, it reduces the input size by 23% compared to JPEG with similar (0.38% lower than GRACE) inference accuracy, while achieving 7.5% higher accuracy over JPEG with a commonly used quality level of 75. Compared to lossless PNG, GRACE saves 80% of the bandwidth with only 0.2% of accuracy loss. For classification tasks, GRACE reduces the bandwidth consumption by 90% over JPEG with the same accuracy.

To our knowledge, GRACE is the first compression algorithm that can adapt to the DNN's perception. It has great potential in the inference-at-edge scenarios. GRACE adds no extra running overhead to existing codecs as it replaces only the compression parameters tailored for human eyes with the ones optimized for the DNN. Its simplicity and backward compatibility ease the deployment. The main contributions of this paper can be summarized as follows:

- We analyze the perception model of the DNNs by measuring the DNN's gradient of loss *w.r.t.* different frequency- and color- components of the input;
- We propose GRACE, a novel compression algorithm that leverages the perception model of a DNN to compress its input without compromising inference accuracy;
- We prototype GRACE based on the JPEG codec without adding extra running overhead, and our thorough evaluation demonstrates its superior performance over prevailing codecs like JPEG, PNG, and WebP.

2 A PRIMER ON IMAGE/VIDEO CODEC

Almost all existing image/video codecs are designed for human, *i.e.*, they compress following the human perception model so that the compression artifacts are mostly invisible.

A typical encoder has the workflow in Fig. 2: *(i)* The (optional) first step is to convert the RGB image to the YUV color space. Since human eyes are insensitive to the chroma signals U & V, the YUV-formatted image is more compressible (§4.4) than its RGB version. The encoder can optionally subsample the U & V signals, which is known as *chroma subsampling*. *(ii)* The encoder then uses the discrete cosine transform

[1]Fig. 1 shows the perceptual sensitivity *w.r.t.* spatial frequencies of a DNN (DRN-D-22 [12], measurement setup follows §4.3.) and that of the human eyes (element-wise reciprocal of the JPEG quantization table [44]).

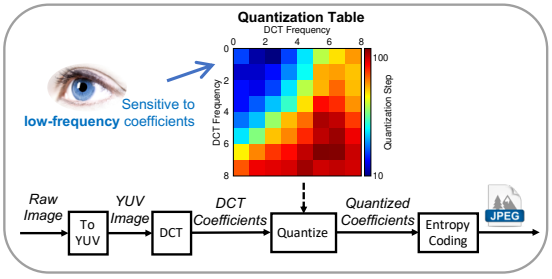


Figure 2: Workflow of a typical image encoder.

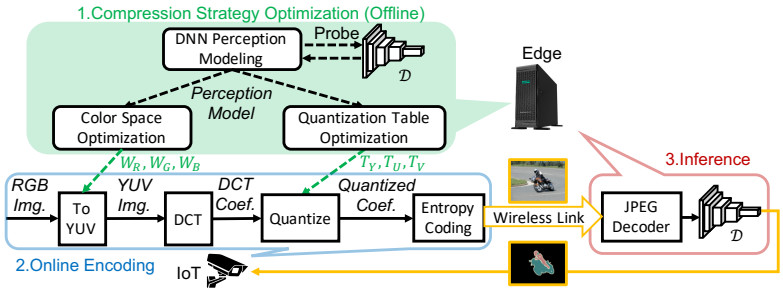


Figure 3: System model.

(DCT) to convert the image to its spatial-frequency representation. DCT helps concentrate most information in a few low-frequency coefficients. *(iii)* The next step is to quantize the DCT coefficients with a *quantization table* $T = \{q_1, ..., q_N\}$ where each q_i is the quantization step on the i-th frequency bin. The design of T ensures quantization artifacts with little human perceptual significance, *e.g.*, human eyes are more sensitive to the low-frequency DCT coefficients, and thus T has smaller quantization steps on these frequency bins. Similarly, the U & V channels have larger quantization steps than the Y channel. Many DCT coefficients (especially at high-frequency bins) become 0 after quantization. Following a Zig-Zag scanning order from low to high frequency, most 0-valued coefficients are chained together and form longer sequences of consecutive 0s. *(iv)* Finally, entropy encoding like run length encoding and Huffman coding leverages such consecutive 0s for further compression.

3 SYSTEM MODEL

This paper considers the inference-at-edge architecture where IoT devices, like drones or surveillance cameras, stream video/image source to DNNs running in the edge for inference, and then, the edge sends the inference results back to the devices. We use *compression target* $\mathcal{D}$ to denote the DNN deployed in the edge, while *compression strategy* $\mathcal{P}$ denotes the set of parameters used by the encoder on the IoT device. GRACE customizes $\mathcal{P}$ for $\mathcal{D}$ in order to save wireless bandwidth without compromising the inference accuracy. $\mathcal{P}$ consists of *(1)* the quantization tables T_Y, T_U, T_V for DCT coefficients on the YUV channels (§5.1), and *(2)* the RGB-to-YUV conversion weights W_R, W_G and W_B (§5.2).

As depicted in Fig. 3, given the compression target $\mathcal{D}$, the edge server (or another server who knows $\mathcal{D}$) can estimate the optimal compression strategy $\mathcal{P}$ for $\mathcal{D}$ via offline optimization. The server then broadcasts $\mathcal{P}$ to IoT devices. Since $\mathcal{P}$ is a small set of constant encoder parameters (only 259 numbers for our prototype in §6) for a given $\mathcal{D}$, broadcasting $\mathcal{P}$ causes negligible overhead. Then the IoT devices follow the optimal encoder parameters in $\mathcal{P}$ to compress the video/image source before streaming them to the edge.

The online compression reuses the existing encoder framework in § 2 (JPEG encoder for our prototype), and GRACE only tunes the encoder parameters to the ones optimized for $\mathcal{D}$. In short, GRACE outperforms JPEG's compression strategy without causing any extra computation overhead. It needs only one-time offline processing to customize the compression strategy in advance for the desired compression target.

In its core, GRACE is a novel algorithm that designs the compression strategy $\mathcal{P}$ for the compression target $\mathcal{D}$. In what follows, we first introduce how to estimate the perception model of $\mathcal{D}$ in §4.3 and §4.4, then we elaborate the optimization algorithms that customize the compression strategies per the estimated perception model in §5.1 and §5.2.

4 UNDERSTANDING DNN'S PERCEPTION

Existing codec designs follow human perception: *(i)* Human eyes are more sensitive to the information with low spatial frequency, so codecs use more bits to represent such information. *(ii)* Human eyes have higher acuity to the brightness than color, so codecs apply more compression on the color information. In this section, we investigate how to measure such perceptual sensitivity of a DNN-based CV system.

4.1 A Primer on DNNs

We first briefly introduce some DNN-related concepts. *Training* is the phase where the DNN learns from the training data. It consists of *(1) a forward pass* to predict the output and estimate the loss L *w.r.t.* the ground truth, and *(2) a backward propagation* to compute the gradients of loss L *w.r.t.* the DNN *weights* and update the weights accordingly. *Inference* is the phase where a trained DNN predicts the output based on the real-world input data by solely a *forward pass*.

Since GRACE compresses for the pre-trained DNN deployed at the edge, we focus on the *inference*. However, since GRACE uses the gradient *w.r.t.* the input to characterize the DNN's perception, we still perform backward propagation after the forward pass to compute the gradients *w.r.t.* the input, without changing the pre-trained DNN weights.

4.2 DNN Gradient & Input Sensitivity

We propose to use a DNN's gradient *w.r.t.* its input as the indicator of its perceptual sensitivity. For a DNN with loss function L and input image of N pixels, *i.e.*, a vector of pixels $\boldsymbol{x} = \{x_1, x_2, ..., x_N\}$, its gradient g_i *w.r.t.* the input pixel x_i is the partial derivative of loss L *w.r.t.* x_i, namely, $g_i = \nabla_{x_i} L = \frac{\partial L}{\partial x_i}$. According to the definition of gradient, when a small Δx_i is added to the input pixel x_i, the DNN loss changes by $\Delta L = g_i \cdot \Delta x_i$. In other words, the gradient characterizes how much a small noise in the input would affect the loss. Since compression can be viewed as adding noise to the image, let e_i denote the compression noise on pixel x_i, it causes a loss change of $\Delta L_i = g_i e_i$, and the total loss change caused by image compression can be modeled as $\Delta L = \sum_{i=1}^{N} g_i e_i$. If a pixel x_i has a high gradient amplitude $\|g_i\|$, even a small noise e_i may cause a high loss increase and significant DNN accuracy drop, *i.e.*, the DNN is more "sensitive" to pixels with higher gradient amplitude. In this way, the gradient *w.r.t.* the DNN input captures the DNN's sensitivity to the input noise.

Ideally, given the pixel-level gradient $\nabla_{\boldsymbol{x}} L$ (*i.e.*, the *sensitivity map* in the spatial domain) for image $\boldsymbol{x}$, we can optimally compress this image but customizing e_i for g_i. However, $\nabla_{\boldsymbol{x}} L$ depends on the image content, and we need to compute $\nabla_{\boldsymbol{x}} L$ for each image $\boldsymbol{x}$ to compress, which is more compute-intensive than the DNN inference due to an additional backward propagation. Note that the DNN-aware compression helps offload the inference workload to the edge, if the compression itself incurs heavier workload than the inference, it is useless in the inference-at-edge scenario.

4.3 Sensitivity *w.r.t.* Spatial Frequencies

To overcome the limitation of the image-specific sensitivity map in the spatial domain, we investigate the DNN's perceptual sensitivity from the frequency domain. Existing codecs like JPEG use DCT-based perception models to characterize the human eyes' sensitivity *w.r.t.* spatial frequencies, and this model is independent of the input images thanks to the translation invariant feature of DCT, *i.e.*, moving the spatial location of an object in the image does not alter its spatial frequency representation. As a result, human eyes are always more sensitive to the low-frequency DCT coefficients in an arbitrary image. Following a similar principle, we use DNN's gradient *w.r.t.* the DCT coefficients (rather than the pixel-level gradient) to model its perceptual sensitivity.

More specifically, given a test image $\boldsymbol{x}$ (used to probe the DNN's sensitivity *w.r.t.* spatial frequencies, but this sensitivity is independent of $\boldsymbol{x}$), we denote its DCT coefficients as $\boldsymbol{s}$ and the DNN's gradient *w.r.t.* DCT coefficients as $\boldsymbol{g} = \nabla_{\boldsymbol{s}} L$. GRACE follows the steps below (Fig. 4) to compute $\boldsymbol{g}$:

(1) transform $\boldsymbol{s}$ to the pixels $\boldsymbol{x}$ using Inverse DCT (IDCT),

(2) feed $\boldsymbol{x}$ to the DNN for inference and get loss L,

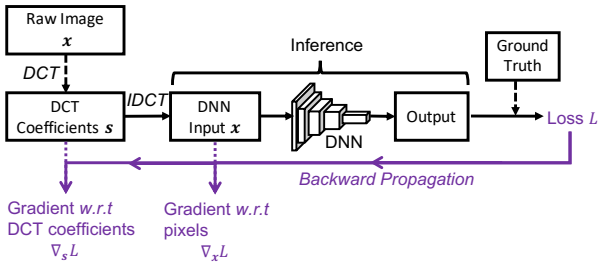


Figure 4: Measuring a DNN's gradient *w.r.t.* its input.

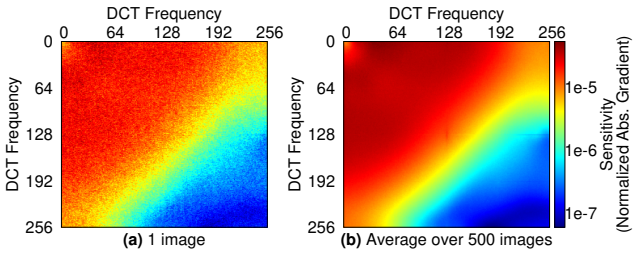


Figure 5: The DCT spectral sensitivity map (gradient amplitude) estimated from 1 or 500 images.

(3) perform backward propagation through the DNN to measure the gradient $\nabla_{\boldsymbol{x}} L$ *w.r.t.* the DNN's direct input $\boldsymbol{x}$,

(4) perform one more step of backward propagation through the Inverse DCT (from $\boldsymbol{x}$ back to $\boldsymbol{s}$) to get $\boldsymbol{g} = \nabla_{\boldsymbol{s}} L$.

After probing the DNN with a set of images to get $\boldsymbol{g}_j$ for each probing image $\boldsymbol{x}_j$, we use the gradient amplitude $\|\boldsymbol{g}\| = \{\|g_1\|, \ldots, \|g_N\|\}$ averaged across all images as the DNN's spectral sensitivity map. (We also record the amplitude of DCT coefficients $\|\boldsymbol{s}\| = \{\|s_1\|, \ldots, \|s_N\|\}$ averaged across all images for the design in §5.1.)

Following the above steps, we model the spectral sensitivity map of a pre-trained DNN for semantic segmentation (DRN-D-38 [48, 49]), with the probing images from the Cityscapes dataset. Fig. 5 shows the spectral sensitivity maps measured from 1 image or averaged over 500 images. We see similar sensitivity distribution across the DCT spectrum for both cases because the spectral sensitivity is the DNN's feature and independent of test images. Therefore, we can use a small image set $\mathcal{I}$ to probe an accurate perception model, which can guide the compression for another image set $\mathcal{T}$ when using the same DNN for inference (§6).

4.4 Sensitivity *w.r.t.* Colors

It is also well-known that human eyes have different perceptual sensitivity to different colors. In what follows, we first introduce how existing codecs leverage such color sensitivity, and then investigate the color sensitivity of DNNs.

4.4.1 Color sensitivity of human and YUV color space. The human retina has three types of color-sensitive cone cells to detect red, green, and blue colors. Existing works reveal that human eyes are most sensitive to the green light with 555 nm

wavelength under the average ambient light level, and least sensitive to the blue light as only 2% of the cone cells are blue-sensitive [39]. Codecs leverage human's color sensitivity to compress with little human perception loss. YUV is a widely used color model and is adopted by popular codecs like *JPEG* and *H.264*. The YUV color space consists of one luminance (brightness) channel Y, and two chrominance channels U & V. Following the human perception model, conversion from RGB to YUV format concentrates most salient information to the Y channel, so that the other two channels (U and V) allow aggressive compression.

Let $R \in [0,1]$, $G \in [0,1]$, and $B \in [0,1]$ denote the value of a pixel in an RGB image, while Y, U, and V denote the corresponding YUV pixel values. According to the CCIR.601 standard [37], the conversion from RGB to YUV is:

$$\begin{cases} Y &= W_R R + W_G G + W_B B \\ U &= \frac{1}{2} \cdot \frac{B-Y}{1-W_B} \\ V &= \frac{1}{2} \cdot \frac{R-Y}{1-W_R} \end{cases} \quad (1)$$

The luminance signal (Y) is the weighted sum of the R, G, and B signals, which is essentially a greyscale image. The selection of weights W_R, W_G, and W_B follows the color sensitivity of human [11]. In CCIR.601 [37], $W_R = 0.299$, $W_G = 0.587$, and $W_B = 0.114$. The physical meaning of these weights is intuitive: When the pixel is saturated green, its RGB value is $(0, 1, 0)$ and the perceived brightness Y is $W_G = 0.587$, while the perceived brightness is $W_B = 0.114$ for a saturated blue pixel with RGB value $(0, 0, 1)$. In other words, for the green and blue pixels of equal power, human eyes perceive the green pixel as brighter, and the weights capture such perceived brightness of certain color. Since $W_R + W_G + W_B = 1$, the RGB-to-YUV conversion does not change the dynamic range of signals, *e.g.*, the Y range is still $[0, 1]$. The physical meaning of the U & V channels is the color difference (scaled to the range $[-0.5, 0.5]$). U is the blue-difference, and V is the red-difference, *i.e.*, When the U & V signals are 0, the color image reduces to a greyscale image. The U & V signals allow aggressive compression with little human perception loss as human eyes have low acuity for them.

4.4.2 Gradient-based color sensitivity. We then investigate the DNN's color sensitivity from the perspective of gradients. Since a typical DNN takes RGB image as input, following similar steps in §4.3, we obtain the gradient $\boldsymbol{g}_R$, $\boldsymbol{g}_G$, and $\boldsymbol{g}_B$ *w.r.t.* the R, G, and B channels. Then for each color channel, we sum the gradient amplitude of all frequency coefficients as our measure of the DNN's sensitivity to this color. In this way, we compare the color sensitivity of DNN and human eyes. The experiment setup follows that in § 4.3. We test 6 DNN models further introduced in §7. The color sensitivity of human is from CCIR.601 [37] – 0.299 for red, 0.587 for green, and 0.114 for blue. Fig. 6 shows different color sensitivity between the DNNs and human eyes, *e.g.*, all tested DNNs show 2× blue-channel sensitivity compared to human because human eyes have fewer blue-sensing cone cells while CV has no such limitation. We also observe different color sensitivity across the DNN models, which necessitates the color space customization when targeting a specific DNN.

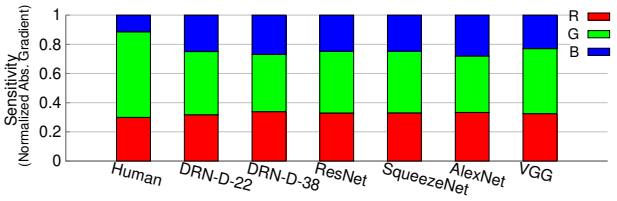


Figure 6: Color sensitivity of DNNs and human.

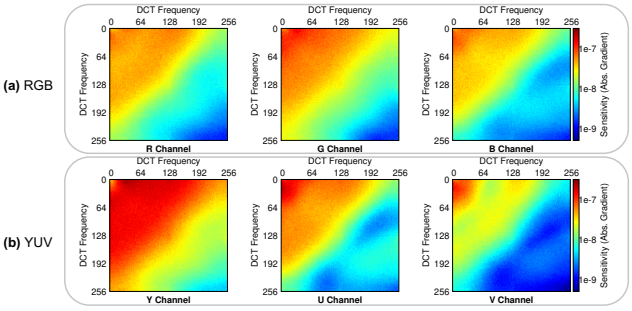


Figure 7: Comparing the DNN's sensitivity map under RGB and YUV formats. YUV concentrates sensitive information to one channel (Y).

4.4.3 Understanding how YUV works from the perspective of gradient. The YUV color space concentrates the sensitive information to the Y channel, and thus the U and V signals can be compressed aggressively. To understand this from the perspective of the gradient, we feed the YUV-formatted `Cityscapes` dataset to the target DNN (`DRN-D-22`) and measure the gradient *w.r.t.* the Y, U, and V channels following §4.3. As shown in Fig. 7, the amplitude of gradient *w.r.t.* the Y channel is higher than U & V channels, which intuitively reveals how the for RGB-to-YUV conversion concentrates the sensitivity information to a single channel. However, since the conversion weights W_R, W_G, and W_B are designed for human eyes rather than DNNs, we still see some sensitive information in the U channel in Fig. 7b. As shown in §4.1, `GRACE` can customize these weights for the target DNN to create more "compressible" U & V channels.

5 LEVERAGING DNN'S PERCEPTION

In this section, we describe how our `GRACE` algorithm leverages the target DNN's perception model *w.r.t.* both the spatial frequencies and colors of the source image input to design a compression strategy that minimizes the file size, with bounded DNN perception loss.

5.1 Leveraging the DNN's Perceptual Sensitivity *w.r.t.* Spatial Frequencies

Existing codecs like JPEG and MPEG quantize the DCT representation of the image/frame by the quantization table designed for the human eyes. GRACE adopts the same codec framework. However, it replaces the human-centric quantization table with a customized one for the DNN, and minimizes the image size without compromising the inference accuracy.

Consider a N-pixel image $\boldsymbol{x}$ (a random vector) as the DNN input. $\boldsymbol{s} = \{s_1, \ldots, s_N\}$ denotes its DCT representation, and q_n denotes the quantization step for the n-th frequency bin. Then, the quantization error $e_n = s_n - \lfloor s_n/q_n \rceil$ and $\|e_n\| \leq q_n/2$. Since DCT concentrates energy on the low frequency, some high-frequency bins may have small $\|s_n\| \leq 2q_n$, where further increasing q_n does not affect the quantization result (if $\|s_n/q_n\| < 0.5$, always $\lfloor s_n/q_n \rceil = 0$). Thus, we bound q_n as Eq. (2) without loss of the generality.

$$0 \leq q_n \leq 2\|s_n\| \tag{2}$$

Our goal is to design the quantization table $T = \{q_1, \ldots, q_N\}$ given the gradient $\boldsymbol{g} = \{g_1, \ldots, g_N\}$ *w.r.t.* spatial frequencies, so that the source size is minimized, the quantization noise (e_n) of each DCT coefficient causes only a small loss change ($g_n e_n$), and the total loss changes of all frequency bins is upperbounded by a constant:

$$\min_{q} \sum_{n=1}^{N} \log_2 \left\| \frac{s_n}{q_n} \right\| \quad \text{s.t.} \quad \sum_{n=1}^{N} g_n e_n \leq C \tag{3}$$

Since the signs of g_n and e_n are random (§4.3), we consider the worst case where they have the same sign (loss increase). Let $d_n = \|g_n\| q_n/2$ denote this worst-case loss increase, from Eq. (2) we have $d_n \leq \theta_n$ where $\theta_n = \|g_n\| \cdot \|s_n\|$ is the upperbound of the loss increase on the n-th frequency bin. The value of θ_n is known as we have obtained $\|g_n\|$ and $\|s_n\|$ following §4.3. To guarantee the worst-case DNN performance, we upperbound the total loss increase as $\sum_{n=1}^{N} d_n \leq B$, where the constant B is the upperbound of loss increase on each DCT block. Although we aim to minimize the file size for a given B, the choice of B depends on the use case. The role of B in GRACE is to control the size-accuracy tradeoff, similar to the *quality level* [44] in JPEG. We show how to balance this tradeoff by tuning B in §7.1.3.

Since s_n does not depend on the quantization step q_n to optimize, we simplify the object function in Eq. (3) following:

$$\arg\min_{q} \sum_{n=1}^{N} \log_2 \left\| \frac{s_n}{q_n} \right\| = \arg\max_{d} \prod_{n=1}^{N} d_n \tag{4}$$

Then the optimization problem in Eq. (3) becomes:

$$\max_{d} \quad \prod_{n=1}^{N} d_n \tag{5a}$$

$$\text{s.t.} \quad \sum_{n=1}^{N} d_n \leq B \tag{5b}$$

$$0 \leq d_n \leq \theta_n \tag{5c}$$

Algorithm 1 Quantization Table Optimization.

Input: $\boldsymbol{g} = \{g_1, \ldots, g_N\}$ – Gradient *w.r.t.* spatial frequencies
$\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_N\}$ – Maximum possible loss increase
B – Configurable upperbound of the allowed loss increase
Output: $T = \{q_1, \ldots, q_N\}$ –Optimal quantization table

$k = N$
while $k > 0$ **do**
  $\overline{d_k} = \frac{B - \sum_{i=k}^{N} \theta_i}{k-1}$
  if $\theta_k < \overline{d_k}$ and $\theta_{k-1} \geq \overline{d_k}$ **then**
    $K = k$
    break
  $k = k - 1$
$\forall n \in [1, N]$, $d_n = \begin{cases} \theta_n & \text{if } n \geq K \\ \frac{B - \sum_{i=K}^{N} \theta_i}{K-1} & \text{otherwise} \end{cases}$, then $q_n = \frac{2d_n}{\|g_n\|}$
$T = \{q_1, \ldots, q_N\}$

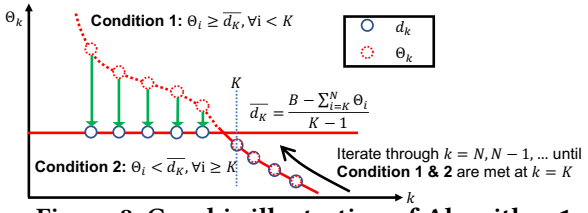


Figure 8: Graphic illustration of Algorithm 1.

With only constraint (5b), the solution of (5a) is $d_n^* = B/N, \forall n \in [1, N]$ based on the Arithmetic Mean-Geometric Mean (AM-GM) inequality [2], but this optimal d_n^* may not be reached if $\theta_n < B/N$ for some frequency bins under constraint (5c), yielding the solution in Theorem 1 below. The proof of this theorem is in the Appendix (A.1).

Theorem 1. *The optimal solution of problem* (5a) *given one or more frequency bins with* $\theta_n < B/N$ *is:*

$$d_n = \begin{cases} \theta_n & \text{if } n \geq K \\ \frac{B - \sum_{i=K}^{N} \theta_i}{K-1} & \text{otherwise} \end{cases} \tag{6}$$

where θ_n *is sorted in descending order,* K *is the highest number that satisfies* $\theta_K < \frac{B - \sum_{i=K+1}^{N} \theta_i}{K}$, $\theta_{K-1} \geq \frac{B - \sum_{i=K}^{N} \theta_i}{K-1}$

After obtaining the optimal d_n for each frequency bin n following Theorem 1, we get the optimal quantization table $T = \{q_1, \ldots, q_N\}$ following $q_n = 2d_n/\|g_n\|$.

Following the above steps, we summarize the quantization optimization algorithm in Alg. 1 and Fig. 8.

5.2 Leveraging the DNN's Color Sensitivity

Existing codecs also exploit the color sensitivity of human eyes by using the YUV color space. Since DNNs have very different color perception from human eyes, we redesign the weights $\{W_R, W_G, W_B\}$ which uniquely define the YUV color space (§4.4), so that the compression on the U and V channels causes minimum perception loss for the target DNN.

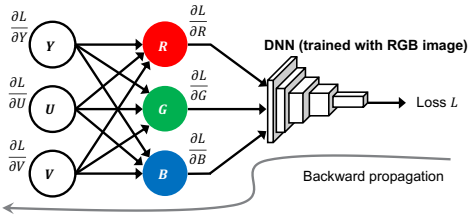
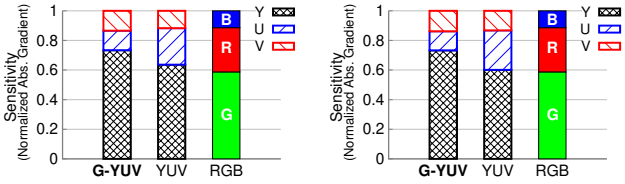


Figure 9: Estimating the gradient *w.r.t.* the YUV channels based on the gradient *w.r.t.* the RGB channels.



(a) Target: DRN-D-22 **(b) Target: DRN-D-38**

Figure 10: G-YUV concentrates salient information to Y channel, making U & V channels more compressible.

More specifically, GRACE characterizes the DNN's perceptual sensitivity to each input channel by the gradient *w.r.t.* this channel. Let $\boldsymbol{g}_Y = \frac{\partial L}{\partial Y}$, $\boldsymbol{g}_U = \frac{\partial L}{\partial U}$, and $\boldsymbol{g}_V = \frac{\partial L}{\partial V}$ denote the gradients *w.r.t.* the Y, U, and V signals of an arbitrary DCT frequency coefficient. Then we minimize the average gradient amplitude $\overline{\boldsymbol{g}_U} = \sum_{i=1}^{N} \|g_U^i\|/N$ of the U channel and $\overline{\boldsymbol{g}_V} = \sum_{i=1}^{N} \|g_V^i\|/N$ of the V channel, which captures the DNN's perceptual sensitivity to the U and V channels.

Since DNNs are typically trained with the RGB images, their inputs need to be converted to the RGB format (Fig. 9). For a target DNN, we can represent its gradients $\{\boldsymbol{g}_Y, \boldsymbol{g}_U, \boldsymbol{g}_V\}$ *w.r.t.* the YUV signals by its gradients $\{\boldsymbol{g}_R, \boldsymbol{g}_G, \boldsymbol{g}_B\}$ *w.r.t.* the RGB signals and the weights $\{W_R, W_G, W_B\}$, then optimize $\{W_R, W_G, W_B\}$ to minimize $\overline{\boldsymbol{g}_U}$ and $\overline{\boldsymbol{g}_V}$. We provide the optimal solution in Theorem 2 and a rigorous proof in A.2.

THEOREM 2. *The optimal weights W_R, W_G, and W_B of the RGB-to-YUV conversion that minimizes $\overline{\boldsymbol{g}_U}$ and $\overline{\boldsymbol{g}_V}$ are:*

$$\begin{cases} W_R &= z_2/(1+z_1+z_2) \\ W_G &= 1/(1+z_1+z_2) \\ W_B &= z_1/(1+z_1+z_2) \end{cases} \tag{7}$$

where $z_1 = \widetilde{\boldsymbol{g}_B/\boldsymbol{g}_G}$ and $z_2 = \widetilde{\boldsymbol{g}_R/\boldsymbol{g}_G}$, the tilde stands for median over all DCT coefficients (of all probing images). $\boldsymbol{g}_R = \frac{\partial L}{\partial R}$, $\boldsymbol{g}_G = \frac{\partial L}{\partial G}$, and $\boldsymbol{g}_B = \frac{\partial L}{\partial B}$ are the target DNN's gradient w.r.t. DCT coefficients on the R, G, and B channel (§4.4).

Since the optimal weights in Eq. (7) are based on the gradients, we name this color space *gradient-driven YUV* or G-YUV. Following the experiment setup in §7.1.5, the results in Fig.10 show the effectiveness of G-YUV. Compared to the conventional YUV or RGB color space, G-YUV concentrates more salient information on the Y channel while suppressing the saliency of the U & V channels.

Algorithm 2 DNN-Aware Codec Parameter Optimization.

Input: $\mathcal{D}$ – Target DNN
B – Configurable upperbound of the allowed loss increase
$\mathcal{I}_{RGB}$ – Image set (RGB) used for DNN sensitivity probing
Output: T_Y, T_U, T_V – The optimal quantization Tables
W_R, W_G, W_B – The optimal RGB-to-YUV conversion weights

1: /***Get weights of G-YUV color space***/
2: Covert $\mathcal{I}_{RGB}$ to spatial frequency domain as $\mathcal{F}_{RGB}$ by DCT
3: Estimate gradients $\boldsymbol{g}_R, \boldsymbol{g}_G, \boldsymbol{g}_B$ of $\mathcal{D}$ to $\mathcal{F}_{RGB}$ following §4.3
4: Compute W_R, W_G, W_B based on $\boldsymbol{g}_R, \boldsymbol{g}_G, \boldsymbol{g}_B$ following Eq. (7)
5: /***Sensitivity evaluation on G-YUV channels***/
6: Convert $\mathcal{I}_{RGB}$ to G-YUV format $\mathcal{I}_{YUV}$ following Eq. (1)
7: Covert $\mathcal{I}_{YUV}$ to spatial frequency domain as $\mathcal{F}_{YUV}$ by DCT
8: Estimate the gradients $\boldsymbol{g}_Y, \boldsymbol{g}_U, \boldsymbol{g}_V$ of $\mathcal{D}$ to $\mathcal{F}_{YUV}$ following §4.3
9: /***Get quantization table for each G-YUV channel***/
10: Compute T_Y, T_U, T_V based on $\boldsymbol{g}_Y, \boldsymbol{g}_U, \boldsymbol{g}_V$ and B as in Alg. 1
11: Broadcast $T_Y, T_U, T_V, W_R, W_G, W_B$ to the clients

5.3 DNN-Aware Compression

Given the modules that leverage the DNN's perceptual sensitivity to spatial frequencies (§5.1) and colors (§5.2), Algorithm 2 further describes how these modules cooperate to achieve DNN-aware compression.

At the server side, it first evaluates the DNN's spatial sensitivity on the RGB channels (line 2 to 3), computes the conversion weights from RGB to G-YUV (W_R, W_G, W_B) based on the RGB sensitivity (line 4), and evaluates the DNN's spatial sensitivity on the G-YUV channels (line 6 to 8). Finally, it computes the optimal quantization table T based on the DNN's spatial sensitivity on each of the G-YUV channels, and then distributes T, W_R, W_G, W_B to clients (line 9 to 11).

The clients receive these parameters and updates their codec accordingly. All steps above are executed offline and incur no extra online running overhead.

6 IMPLEMENTATION

Since most existing DNN models only support image as input, we prototype GRACE on the popular JPEG image codec, and leave the DNN-aware video codec for future work. GRACE does not modify DNNs. It only probes a DNN's perception using a few test images, and then provides an optimized compression strategy based on thus-obtained perception model. Our prototype randomly selects 15 images from the original dataset to probe the DNN's perception model offline and uses all other images for the online performance evaluation.

The compression strategy is a set of codec parameters, including the quantization tables and the RGB-to-YUV conversion weights. GRACE reuses the JPEG codec framework, and only tunes the codec parameters following the values optimized for the target DNN. In this way, the images compressed by GRACE encoder can be decoded by vanilla JPEG decoders, achieving good compatibility.

We use Pytorch to implement both the offline compression strategy optimization and online inference. Measuring the DNN's perception model (*i.e.*, the gradients *w.r.t.* the DCT coefficients in different color channels of the input image) is the key of GRACE, and we use the Autograd package of Pytorch to obtain the gradients of the DNN's negative log likelihood loss. Existing codecs like JPEG use block-wise DCT (e.g., 8×8), as opposed to applying DCT to the entire image, and this is also necessary for GRACE. Otherwise, the DCT coefficient of a large image/block can have a high value (especially for low-frequency bands) exceeding the symbol length supported in the JPEG codec framework. Our GRACE prototype segments the image into blocks with size $D \times D$, and then perform $D \times D$ DCT. After getting the block-wise gradients on all blocks, we average them to estimate the DNN's sensitivity across the $D \times D$ spectrum. We test various block sizes from 8 to 128. Our prototype uses $D = 16$ since it yields slightly better performance than other sizes.

7 EVALUATION

GRACE is a generic source coding solution for a wide range of deep learning tasks with differentiable DNNs, and our evaluation covers two essential types of CV tasks – semantic segmentation (§7.1) and image classification (§7.2). Overall, we evaluate GRACE over various conditions, including 2 CV tasks, 2 datasets, 5 benchmark codecs, and 6 DNN models.

7.1 Compression for Segmentation

We first evaluate GRACE over semantic segmentation – one of the most challenging CV tasks. We use the compressed file size and DNN inference accuracy as two performance metrics to capture the size-accuracy tradeoff. Next, we use the mean intersection-over-union (*mIoU*) to measure the DNN inference accuracy. For a particular class, we denote $N(t,p)$ as the number of pixels that belong to, and are predicted as, the class (true-positive), $N(f,n)$ as the number of pixels that belong to the class, but are predicted as other classes (false-negatives), and $N(f,p)$ as the number of pixels that belong to other classes, but are predicted as the class (false-positives). Then $IoU = 100 \cdot N(t,p)/\big(N(t,p) + N(f,n) + N(f,p)\big)$.

If the predicted segmentation exactly matches the ground truth, $IoU = 100$, while IoU= 0 implies no overlap between the predicted and true segmentation. *mIoU* is the average *IoU* value of all classes.

This evaluation employs two Dilated Residual Network models DRN-D-22 & DRN-D-38 [48, 49] for inference using the Cityscapes dataset [7] with high-resolution (2048×1024) PNG images. We set the upperbound B of per-block DNN loss increase (§5.1) to 1.5×10^{-4} for DRN-D-38 and 1.1×10^{-4} for DRN-D-22, which leads to good size-accuracy tradeoff (We show how the choice of B affects the tradeoff in §7.1.3).

7.1.1 Comparison with Prevailing Codecs. Following the experiment settings above, we benchmark GRACE against popular image codecs (including BMP, PNG [4], JPEG [44], the advanced WebP [15] codec from Google), and the H.264 [38] video codec. Table. 1 gives a side-by-side performance comparison of these codecs under their typical configurations (in table footnotes), where the target DNN is the DRN-D-38 model. We record the file size, DNN inference accuracy, the compression ratio, and bit per pixel (BPP) computed based on the file size. We see that GRACE achieves a high compression ratio of 16.89 : 1 *w.r.t.* the uncompressed BMP format with BPP=24 for RGB image, at the cost of a slight inference accuracy loss. On the other hand, PNG does not affect the inference accuracy as it is lossless. However, it suffers from a low compression ratio of 2.66 : 1, consuming 6.34× more bandwidth than GRACE does. We test JPEG with a high quality (Q=95), which consumes 29.37% more bandwidth than GRACE, but it yields even lower inference accuracy than GRACE. For lossy WebP, even at the highest quality (Q=100), the resulting inference accuracy is 0.31% lower than GRACE, while consuming 32.1% more bandwidth over GRACE.

In Fig. 11, we further show one running instance of GRACE to demonstrate its friendliness to the DNN. By comparing the raw PNG image in Fig. 10c and the GRACE-compressed image in Fig. 10d, a human can notice the slight difference that is invisible to the DNN, whereas the DNN inference outputs in Fig. 10e and 10f are almost identical.

Since modern codecs are adjustable to balance the size and quality, in what follows, we compare GRACE with JPEG, WebP and H.264 under various configurations.

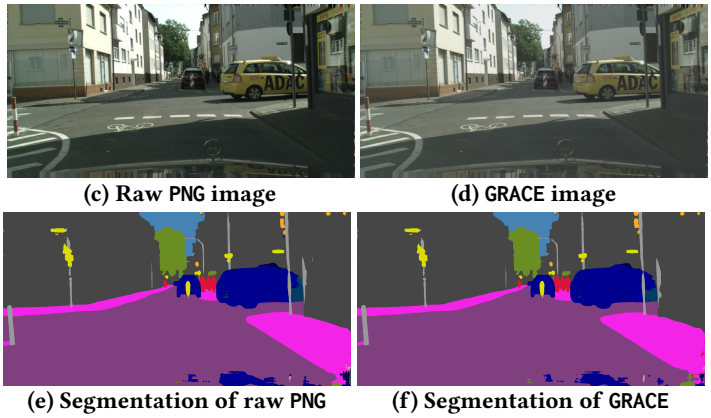
GRACE vs. JPEG. Since JPEG supports quality levels ranging from 1 to 100 [44], we transform the original Cityscapes dataset with the lossless PNG format to JPEG-formatted copies with quality level 75, 85, 95, and 100. From both Fig. 12a and 12b, we see that JPEG causes high accuracy drop at JPEG quality level of 85 and 75. The accuracy dropped by at most 7.43% (Q=75 on DRN-D-22) and at least 2.03% (Q=85 on DRN-D-38). In other words, the commonly used JPEG with medium/low quality is unsuitable for the inference of semantic segmentation. We then look at JPEG with high quality (Q=95), which performs strictly worse than GRACE over both DRN-D-38 and DRN-D-22. Over DRN-D-38, JPEG yields 29.37% larger file size than GRACE, while its accuracy becomes lower than GRACE by 0.35%. Similarly, it yields 39.88% larger file size and 0.23% lower accuracy than GRACE, over DRN-D-22.

GRACE vs. WebP. WebP also supports the quality levels as in JPEG. In this experiment, we convert the Cityscapes dataset to multiple WebP-formatted copies, each with a particular quality. As shown in Fig. 12c, lossless WebP has an average file size of 1.8MB, which does not fit bandwidth-constraint applications. For the lossy WebP, the highest quality with

Comp. Format	Mean Size(KB)	Accuracy (mIoU)	Comp. Ratio	BPP
GRACE	370.82	70.84%	16.97 : 1	1.41
JPEG[a]	481.97	70.47%	13.05 : 1	1.84
WebP[b]	491.01	70.53%	12.81 : 1	1.87
H.264[c]	369.25	70.38%	17.04 : 1	1.41
PNG	2362.81	71.35%	2.66 : 1	9.01
BMP	6291.46	71.35%	1 : 1	24.00

Table 1: File size & inference accuracy of GRACE and other popular formats.

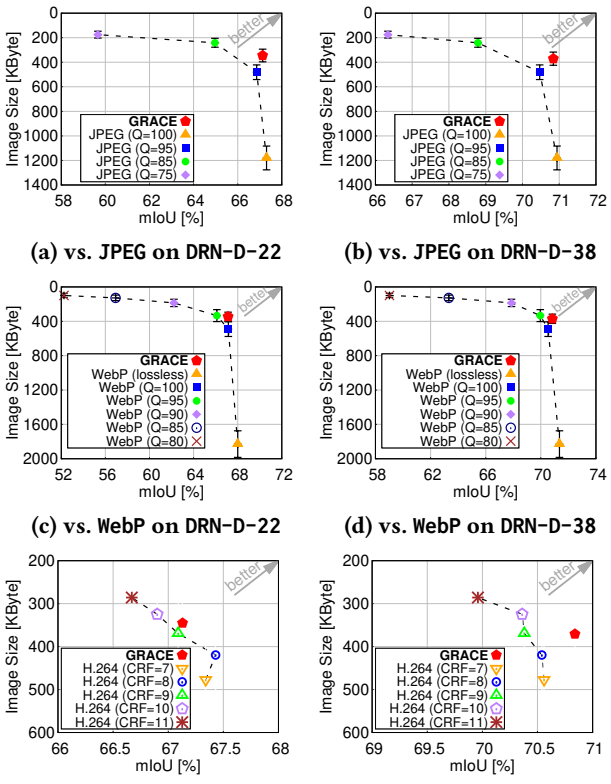
[a]Q=95
[b]Q=100
[c]CRF=9



(c) Raw PNG image **(d) GRACE image**

(e) Segmentation of raw PNG **(f) Segmentation of GRACE**

Figure 11: Comparing the semantic segmentation result with PNG



(a) vs. JPEG on DRN-D-22 **(b) vs. JPEG on DRN-D-38**

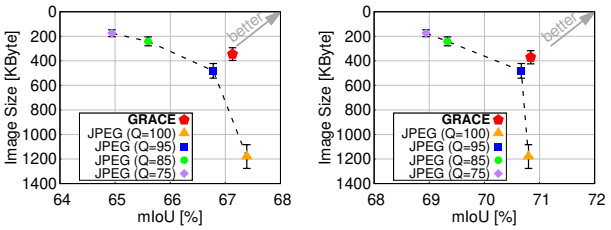
(c) vs. WebP on DRN-D-22 **(d) vs. WebP on DRN-D-38**

(e) vs. H.264 on DRN-D-22 **(f) vs. H.264 on DRN-D-38**

Figure 12: The file size & inference accuracy (mIoU) of GRACE and benchmark codecs.

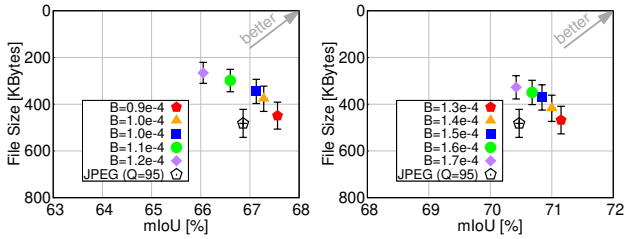
Q=100 causes 42% larger file size over GRACE, but the accuracy is almost identical (for DRN-D-22) or even lower (for DRN-D-38). We also see that WebP of Q=95 yields a similar file size but a 1% lower accuracy compared to GRACE. Other lower quality levels cause more severe accuracy losses. Although GRACE outperforms WebP in the above experiment, the setup slightly underestimates GRACE. In addition to quantization optimization, WebP uses intra-frame prediction to further improve the compression [15]. On the other hand, GRACE prototype only showcases DNN-aware quantization and does not implement intra-frame prediction yet. Furthermore, WebP employs boolean arithmetic coding for entropy encoding, which has a higher compression ratio than the Huffman coding used in JPEG/GRACE. Later, GRACE can also incorporate such advanced components as intra-frame prediction and arithmetic coding for further improvement. Finally, WebP encoder runs slowly at only 4.42 FPS in our test (for Q=95), around 3 times slower than JPEG/GRACE (running at 15.02 FPS under similar compression ratio) due to its extra step of intra-frame prediction and different entropy coding algorithm. This high time overhead makes WebP unsuitable for online encoding in the inference-at-edge architecture.

GRACE vs. H.264. The H.264 codec uses the constant rate factor (CRF) to control the video quality. CRF ranges from 0 to 51, and a smaller CRF indicates a higher quality. Like previous experiments, we also create multiple H.264-encoded datasets, each with a particular CRF. Since H.264 is a video codec while the Cityscapes dataset consists of images, we use FFMPEG to merge images into an H.264 video, and then split this video to a new set of images as the compressed dataset. Since H.264 compression leverages the temporal correlation between frames, the total size increases after splitting the video into images, and we use the video size divided by the frame number as the H.264 average frame size. In Fig. 12e and 12f, we report test results for CRF levels of 7, 8, 9, 10, and 11 as other CRFs cause either too large file size or too low accuracy (*e.g.*, the default CRF=23 causes a low accuracy of 44.84%). The results demonstrate GRACE's superior performance over H.264 even the comparison underestimates our solution (H.264 has the inter- and intra-frame prediction while our prototype does not implement that yet). We see that any H.264 configuration with a smaller file size than GRACE suffers from lower accuracy. H.264 with CRF=8 has slightly (0.3%) higher accuracy over GRACE when targeting



(a) DRN-D-22 (b) DRN-D-38

Figure 13: GRACE vs. JPEG when DNNs for JPEG inference are trained by JPEG images with the same quality.



(a) DRN-D-22 (b) DRN-D-38

Figure 14: GRACE balances file size and inference accuracy by controlling the quota of DNN loss increase.

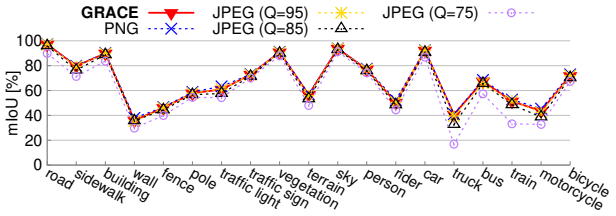


Figure 15: Per-class semantic segmentation accuracy.

7.1.2 Comparison with DNNs trained by compressed images. GRACE optimizes the source compression for DNNs that are trained with uncompressed dataset. One alternative approach would be to train DNNs with compressed dataset, whose quality is same as test images, *e.g.*, if the test images have a JPEG quality of 75, then we train the DNN with the training set of JPEG quality 75. Apparently, this alternative compromises the accuracy, because protecting the salient information during compression (as in GRACE) is always better than discarding the salient information and then trying to compensate by tuning the DNN weights (the alternative). We validate this finding in the experiment below.

We compress the Cityscapes dataset to multiple copies with JPEG quality of 75, 85, 95 and 100, and then do the training and inference on datasets with matching quality. The results are shown in Fig. 13, where the GRACE configuration remains the same as previous experiments. By comparing Fig. 13 and the JPEG performance over DNNs trained with the uncompressed dataset (Fig. 12), we see that training the DNN with the image quality matching the test set improves the accuracy, especially for the low-quality images. For instance, in Fig. 12, the accuracy is 66.34% given the test data of JPEG quality 75 and target DNN (DRN-D-38) trained with the uncompressed dataset, while the accuracy increases to 68.94% when we train the target DNN with the training set of JPEG quality 75, which is a 2.60% improvement. However, GRACE still outperforms JPEG even when we train and test the target DNN with the same JPEG quality. Over DRN-D-38, GRACE achieves 28% smaller file size and 0.17% higher accuracy than the alternative (training & testing the DNN by JPEG quality of 95) with the closest performance.

7.1.3 Adjusting the bound of DNN perception loss in GRACE. GRACE can minimize the file size, as an upperbound B of the DNN perception loss increases (§5.1). By adjusting this upperbound B (*i.e.*, the quota of DNN loss increase), we can control the tradeoff between the file size and inference accuracy for the desired use cases, just like the various quality levels of JPEG. In this experiment, we follow the same setup of the previous one in §7.1.1. Since GRACE adopts 16×16 DCT block (§6), we assign the same loss quota to all the DCT blocks and tune this block-wise loss quota. We also plot the performance of JPEG with Q=95 as a benchmark since it has the closest performance to GRACE among all tested JPEG quality levels. From Fig. 14a and 14b, we observe that GRACE allows us to adjust the tradeoff between the file size an inference accuracy – given a higher quota of loss increase, it compresses more aggressively, yielding small file size at the cost of lower accuracy. Under a proper configuration of B, GRACE easily outperforms JPEG with Q=95.

7.1.4 More details on the inference accuracy and file size. In previous experiments, the image is segmented to 19 classes defined in the Cityscapes dataset, and we use the mean *IoU* over the 19 classes as the inference accuracy. In Fig. 15 we further show the per-class *IoU* from the experiments in §7.1.1. We only plot the results for GRACE over DRN-D-22 due to the space limit. We see that GRACE leads to almost no accuracy loss on every class. JPEG with Q=95 achieves similar accuracy at the cost of much higher file size, while JPEG with lower quality suffers from significant accuracy loss on some classes (24% accuracy loss on class 15 for Q=75). In Fig. 16, we further provide the CDF of the file sizes from the same experiments for GRACE and other formats with similar accuracy. We first see that DRN-D-22 allows more aggressive compression for GRACE than DRN-D-38 as it generally has lower inference accuracy than DRN-D-38 and can "see" less details from the input image. We also see that GRACE and JPEG Q=95 have similar file size variation.

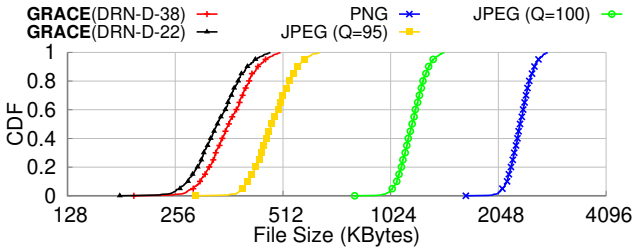
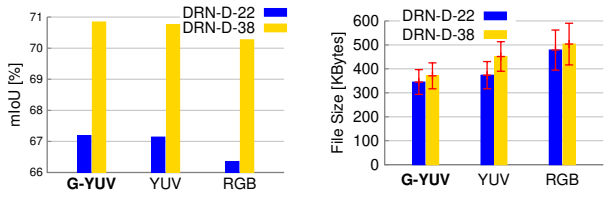


Figure 16: File size comparison between GRACE and benchmark formats with comparable accuracy.



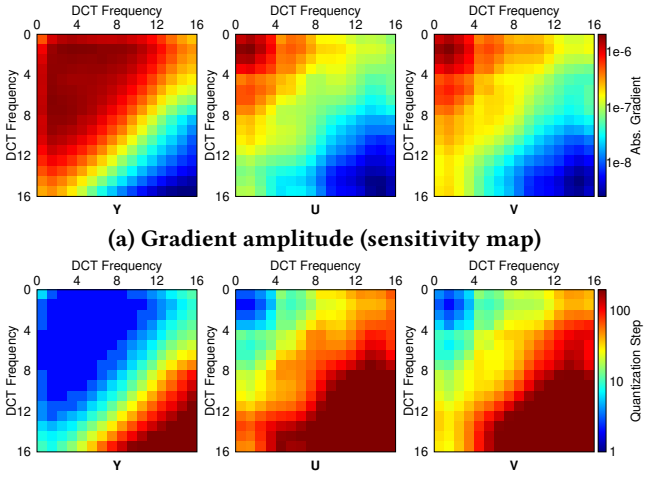
(a) Accuracy (mIoU) **(b) File size**

Figure 17: Using gradient-driven color space (G-YUV) reduces the file size and improves the accuracy.

7.1.5 Validating the G-YUV *color space.* In this experiment, we validate the design of G-YUV color space by showing its performance gain over conventional RGB and YUV color spaces given the same GRACE-customized quantization table. The experimental setting is the same as §7.1.1, except that we convert the images to different color spaces before compression. When used as the color space for GRACE compression, G-YUV leads to slightly higher accuracy than YUV and RGB (Fig. 17a) while achieving the smallest file size (Fig. 17b), *e.g.*, it enables 18% of bandwidth saving compared to YUV under DRN-D-38, without sacrificing the accuracy.

We further show the perception model and quantization table on the 16×16 DCT block in Fig. 18. Since G-YUV yields a Y channel with a high gradient amplitude (Fig. 18a), it leads to less aggressive compression (Fig. 18b) for the salient information on Y channel, thus better protecting such information than RGB or YUV. Meanwhile, G-YUV ensures low gradients on the U & V channels (Fig. 18a), and thus we can use high quantization steps on these channels (Fig. 18b) to produce more consecutive 0s for entropy coding to leverage.

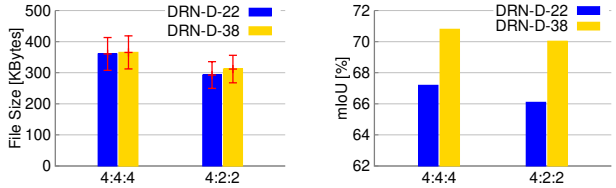
7.1.6 Chroma subsampling. In this experiment, we compare the performance of GRACE configurations using 4:4:4 (no subsampling) and 4:2:2 (halving the horizontal resolution of U and V channels). Fig. 19a shows that 4:2:2 subsampling reduces the image size by 19% for DRN-D-22 and 15% for DRN-D-38. However, this file size reduction is at the cost of 1% accuracy drop as in Fig. 19b. Fig. 18a further shows that the two chroma channels U and V still have high gradients on a small number of low-frequency DCT coefficients,



(a) Gradient amplitude (sensitivity map)

(b) Quantization Table

Figure 18: The DNN perception model and quantization table of the G-YUV color space.



(a) File size **(b) Accuracy (mIoU)**

Figure 19: Chroma subsampling slightly reduces the file size but causes significant accuracy drop.

Comp. Target \ Inference	DRN-D-22	DRN-D-38
DRN-D-22	67.13%	70.65%
DRN-D-38	67.25%	70.84%

Table 2: Cross-model segmentation accuracy.

and chroma subsampling distorts such salient coefficients, causing the accuracy loss. As a result, although chroma subsampling is common in conventional codecs, it is unsuitable when compressing for DNNs.

7.1.7 When compression target is not inference model. Could we use the compression strategy $\mathcal{P}$ customized for one DNN with another DNN for inference? Following the setting in §7.1.1, we test the inference accuracy by sending the images compressed by GRACE to a different DNN for inference. The result in Table. 2 shows a similar accuracy when the compression target differs from the inference model. To understand this, we compare the sensitivity maps of DRN-D-22 and DRN-D-38 in Fig. 20 (only show Y channel due to the space limit), and their similar sensitivity *w.r.t.* spatial frequencies explain the similar accuracy. Meanwhile, we see that DRN-D-22 has a more evenly distributed sensitivity map than DRN-D-38, and thus the compression strategy customized

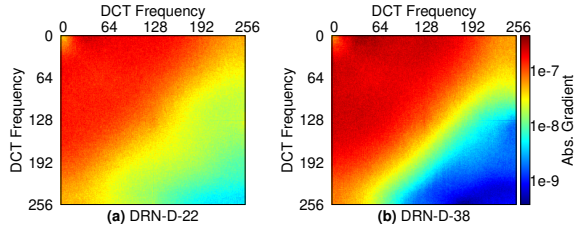


Figure 20: Sensitivity map of DRN-D-22 and DRN-D-38.

Format \ Inference	ResNet	SqueezeNet	AlexNet	VGG
GRACE	12.39 : 1	7.97 : 1	7.42 : 1	4.75 : 1
JPEG (Q=96)	1.26 : 1			
JPEG (Q=85)	3.17 : 1			

(a) Compression Ratio.

Format \ Inference	ResNet	SqueezeNet	AlexNet	VGG
GRACE	96.08%	93.46%	90.85%	96.73%
JPEG (Q=96)	96.08%	93.46%	90.85%	96.73%
JPEG (Q=85)	94.77%	92.16%	90.20%	95.42%

(b) Classification Accuracy.

Table 3: GRACE *v.s.* JPEG over classification tasks.

for DRN-D-22 has less protection for salient information. As a result, when targeting DRN-D-38, the accuracy is always slightly higher regardless of the inference model. A similar cross-model test for image classification is in §7.2.2.

7.2 Compression for Classification

We then evaluate GRACE over another essential CV task – image classification. We use the ImageNet dataset [10] and evaluate GRACE targeting 4 types of widely-used DNNs including SqueezeNet [22], a small DNN designed for mobile clients, and more complicated AlexNet [24], ResNet-18 [18], and VGG-11 [41] as inference-at-edge enables mobile devices to enjoy the high accuracy of complicated DNNs. The format of the original dataset is JPEG of Q=96, and we compress it to JPEG of Q=85 as the other benchmark. For the size-accuracy tradeoff, we adjust the upperbound of DNN loss increase (§7.1.3) to reach the minimum file size that leads to no accuracy loss *w.r.t.* the original JPEG dataset.

7.2.1 GRACE *compression over different DNN models.* Following the experiment setup above, we first compare GRACE and JPEG when they compress for various DNN models that classify images, The results in Table. 3a and 3b show that GRACE outperforms the JPEG configurations, achieving a higher compression ratio than the original JPEG images of Q=96 for all tested DNN models without any loss in the classification accuracy. Meanwhile, JPEG of Q=85 achieves both a lower compression ratio and lower accuracy than GRACE does.

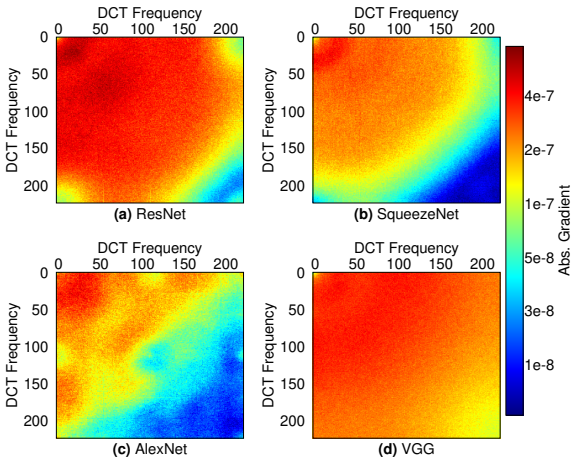


Figure 21: Sensitivity map of DNNs for classification.

Comp. Target \ Inference	ResNet	SqueezeNet	AlexNet	VGG
ResNet	96.08%	92.81%	87.58%	93.46%
SqueezeNet	96.08%	93.46%	88.89%	96.08%
AlexNet	95.42%	93.46%	90.85%	96.08%
VGG	96.08%	93.46%	90.85%	96.73%

Table 4: Cross-model classification accuracy.

We then compare the GRACE performance across different target DNN models. The ranking of compression ratio is ResNet > SqueezeNet > AlexNet > VGG, while the accuracy ranking is VGG > ResNet > SqueezeNet > AlexNet. We further plot (only Y channel) sensitivity maps of these DNNs in Fig. 21: VGG has high sensitivity across the entire spectrum. Hence, GRACE cannot compress much, leading to the lowest compression ratio & the highest inference accuracy. ResNet has high sensitivity on the low-frequency bands, which ensures GRACE to allocate more bits to these sensitive information. Therefore, GRACE for ResNet achieves the highest compression ratio and an accuracy close to VGG. SqueezeNet & AlexNet are insensitive to most frequency bands, and thus suffer from low accuracy, but such large insensitive regions contribute to compression ratios higher than VGG.

7.2.2 When compression target is not inference model. We also perform a cross-model test similar to §7.1.7. Following the experiment setup in §7.2.1, we use the compression strategy $\mathcal{P}$, customized for one DNN, for source compression and another DNN for inference. The result in Table. 4 shows:

When using ResNet for inference, $\mathcal{P}$ for SqueezeNet or VGG leads to no accuracy loss, because ResNet is mostly sensitive to the low-frequency coefficients (Fig. 21a) while SqueezeNet and VGG are also sensitive to such coefficients. The sensitivity map of AlexNet in Fig. 21c shows low-sensitivity "holes" around the low frequencies. Hence, $\mathcal{P}$ for AlexNet discards sensitive information for ResNet, causing 0.56% of accuracy loss.

When using SqueezeNet for inference, $\mathcal{P}$ only for ResNet causes lower accuracy due to its highest compression ratio.

When using AlexNet for inference, $\mathcal{P}$ for ResNet or SqueezeNet yields higher compression ratio, causing accuracy drop. The conservative $\mathcal{P}$ for VGG covers most high-sensitivity regions of AlexNet, leading to no accuracy loss.

When using VGG for inference, $\mathcal{P}$ for any other DNNs causes accuracy loss since VGG is sensitive to most frequency bands. Among them, $\mathcal{P}$ for ResNet causes the highest accuracy loss of 3.27% due to its highest compression ratio.

In sum, if the high-sensitivity frequency bands of model $\mathcal{D}_1$ also covers that of model $\mathcal{D}_2$, using $\mathcal{D}_1$ as compression target and $\mathcal{D}_2$ as inference model leads to small or no accuracy loss. We have a further discussion in §9.

8 RELATED WORKS

Bringing DNN to the edge. Deep learning on IoT has a huge potential market. Both the academia and industry have invested significant efforts in this area. One way to reduce the DNN computational cost is to simplify the DNN by pruning or quantizing the weights [8, 9, 17, 20, 21, 26, 36, 50]. However, such a pruned DNN remains overwhelming for resource-constrained IoT devices.

With the upcoming 5G networks featuring IoT, low latency, and mobile edge computing, a promising direction is to offload the DNN inference to the edge, which draws enormous attention from the IT industry like Hewlett Packard Enterprise [19], Facebook [46], Google [16], Microsoft [32], and Nvidia [33]. Our work tackles the IoT scalability issue caused by the wireless bottleneck in such an inference-at-edge architecture.

Adversarial Attacks. The gradient *w.r.t.* DNN input is the foundation of *adversarial attacks* [3, 5, 13, 14, 25, 29], where an attacker intentionally designs DNN inputs called *adversarial examples* to fool the DNN. For instance, the fast gradient sign attack [14] maximizes the DNN loss following the sign of the gradient *w.r.t.* DNN input – if the gradient *w.r.t.* an input pixel is positive/negative, then a positive/negative noise is added to this pixel to raise the loss.

Compared to adversarial attacks, our work uses the gradient *w.r.t.* DNN input in the opposite way: The gradient guides GRACE to protect the salient information in the DNN input rather than breaking them.

DNN-based compression. From a high level, image compression maps the input from the image space with a larger dimension to the latent space with a smaller dimension, and image decoding does the opposite. DNNs can learn and perform such mappings:

Variational autoencoder (VAE) [6, 23, 31, 43] learns efficient data representation via unsupervised training. A VAE consists of two neural networks: the *encoder* that compresses the input to the latent-space code, and the *decoder* that reconstructs the original image from the code. Such an encoder-decoder architecture can also be used for image interpolation in DNN-based video codecs [47].

Generative adversarial network (GAN) [1, 28, 40] is also a powerful tool for compression. A GAN consists of two neural networks: the *generator* that tries to generate images to mimic the original ones, and the *discriminator* that tries to classify original and generated images. Existing works typically use the GAN generator to decode compressed (*e.g.*, by the VAE encoder) images. We can also compress images with a pre-trained GAN generator by training the input rather than the weights of the generator [28].

Content-aware image compression [27, 35] leverages DNN to understand the image content for better compression, which extracts a saliency map from the input image using a DNN, and then allocates more bits to the salient regions.

Compared to these DNN-based compression works, GRACE has the following advantages:

(i) GRACE is the first work to use the frequency-domain gradient as a DNN's sensitivity map. Compared to the spatial-domain saliency map [27, 35] that depends on the image content, the frequency-domain sensitivity map is a feature of the DNN and independent of the images (Fig. 5). Therefore, GRACE can optimize the codec parameter offline and reuse the simple JPEG codec for online compression, thus achieving much lower computation overhead than DNN-based codecs that perform inference for every image to compress.

(ii) GRACE guarantees DNN inference performance. In contrast, existing codecs aim to achieve either high fidelity (like PSNR or MSE) or good human-perceived quality (like SSIM [45]), without guarantee for the inference accuracy.

(iii) GRACE optimizes the compression under rigorous proof (A.1 and A.2), while the DNN-based solutions rely on heuristics to determine the crucial hyper-parameters.

Compression for DNNs. Due to the lack of understanding on the DNN perception, only a few heuristic works [30] discuss about compression for DNNs. They assume that the contribution of a frequency band to the DNN output depends on the magnitude of its corresponding DCT coefficient. However, this heuristic has several crucial problems:

A high magnitude does not guarantee high contribution to the DNN output. e.g., the DC coefficient of an image typically has the highest magnitude, but does not affect the object shape that depends on the relative values of neighboring pixels. If we view the DNN as a filter, signals on certain frequency bands may not pass even with high magnitude.

A high contribution to the DNN output does not guarantee high DNN sensitivity. Assuming a setup with input DCT coefficients $\boldsymbol{s}$ and the target DNN $\mathcal{D}$, where a particular frequency band j has a high contribution to the DNN output. We can multiply s_j by M to get a new input $\boldsymbol{s}^*$ and append

to $\mathcal{D}$ a first layer that only performs s_j^*/M for the band j, and then we have a new DNN $\mathcal{D}^*$ with input $\boldsymbol{s}^*$. In this new setup, the DNN output remains the same, so band j has a high contribution to $\mathcal{D}^*$. According to the chain rule, the gradient *w.r.t.* s_j^* is $g_j^* = g_j \frac{1}{M}$. If the same noise e_j adds to the signal of band j for both the original and new setup, the change in the loss of $\mathcal{D}^*$ is $\Delta L^* = g_j^* e_n = \Delta L/M$, *i.e.*, adding a noise to band j causes M times smaller loss change which is negligible under large M. Therefore, the DNN can be insensitive to a band with a high contribution to its output. From the information theory perspective, $s_j^* = Ms_j$ requires $\log_2 M$ more bits to represent than s_j, but the amount of information carried on band j (contribution) remains unchanged. Therefore, s_j^* allows more compression with the $\log_2 M$ redundant bits and thus band j has less DNN sensitivity.

The magnitudes of DCT coefficients depend solely on the images but not the DNN. Hence the solution in [30] compresses images in the same way for all CV tasks (*i.e.*, DNNs). However, different tasks need different compression strategies, *e.g.*, image classification allows for more aggressive compression than semantic segmentation, as shown in §7.

9 DISCUSSION AND FUTURE WORK

Generic compression for multiple DNNs. GRACE enables compression strategy customization for each particular DNN-based CV tasks. For instance, image classification allows more aggressive compression than semantic segmentation, and thus when streaming to a server that only performs classification but not semantic segmentation, a customized compression strategy can significantly save the bandwidth.

GRACE can also design the generic compression strategy for multiple DNNs. Assuming a DNN is sensitive to frequency bands F_1, while another one is sensitive to F_2. If the compressed images need to fit both DNNs, the images should have less compression on $F_1 \cup F_2$, which trades compression ratio for generality. More specifically, if the compression needs to fit M different CV tasks, and the sensitivity map of the DNN for the i-th task is $\boldsymbol{g}^i = \{g_1^i, g_2^i, ..., g_k^i, ..., g_N^i\}$, GRACE can use a more generic sensitivity map $\boldsymbol{g^*} = \{g_1^*, g_2^*, ..., g_k^*, ..., g_N^*\}$ with $g_k^* = \max(g_k^1, g_k^2, ..., g_k^i, ..., g_k^M)$ to compute the optimal quantization table following §5.

DNN-and-codec joint optimization. From a high level, there are three options to enable better collaboration between the source codec and DNN: *(1)* Customizing the compression strategy for a given DNN. *(2)* Re-training the DNN for a particular strategy mechanism. *(3)* Optimizing both the DNN and compression strategy jointly. This work follows the first option as the IoT devices cannot change a DNN already deployed at the edge, and §7.1.2 shows that the second option is suboptimal. In future work, we plan to investigate the third option using GAN – source coding acts as the generator while edge inference acts at the discriminator. This option should further improve the performance if the edge operators are willing to update the DNNs at the edge.

Compression for various deep learning tasks. Besides image compression, the concept of gradient-driven compression in GRACE can be used to compress the input of any DNN that is differentiable. For instance, GRACE may potentially optimize the audio compression for DNN-based speech recognition tasks. Unlike prevailing audio encoders (like MP3) that rely on the frequency response of human ears, GRACE can compress the audio signal on different frequency bands following their importance to the speech-recognition DNN rather than their saliency to human ears.

Compression for the training data. This work focuses on compressing the test data. In practice, the vast size of the training dataset is another major problem for deep-learning research, and the principle of GRACE can be applied here: if a DNN cannot learning anything from some particular frequency bands, they can be removed to reduce the training data size. We plan to investigate this as our future work.

10 CONCLUSION

This paper presents GRACE, a DNN-aware source compression algorithm to facilitate the inference at the edge for IoT computer vision. By leveraging the perception model of the target DNN at the edge, GRACE saves the bandwidth without compromising the inference accuracy. More specifically, we propose to model a DNN's perceptual sensitivity by its gradient *w.r.t.* the spatial-frequency bands and colors of the input. Based on this perception model, we design GRACE that formulates the image compression for DNN as a set of optimization problems. GRACE then estimates optimized compression strategy that minimizes the file size with bounded DNN perception loss. We prototype GRACE on the popular JPEG codec, and our system design guarantees that the complexity and running overhead of GRACE is identical to JPEG – no extra overhead. Our evaluation over both semantic segmentation and image classification tasks demonstrate that GRACE significantly outperforms JPEG and PNG, achieving a higher compression ratio with little or no inference accuracy loss. Besides image compression, in future work, we plan to extend the concept of *compression fitting the DNN's perception model* to video and audio codecs.

ACKNOWLEDGMENTS

We appreciate the anonymous reviewers and shepherd for their insightful comments.

REFERENCES

[1] Eirikur Agustsson, Michael Tschannen, Fabian Mentzer, Radu Timofte, and Luc Van Gool. 2018. Generative Adversarial Networks for Extreme Learned Image Compression. *arXiv preprint arXiv:1804.02958* (2018).

[2] Denise Arnold and Graham Arnold. 1993. Four Unit Mathematics.

[3] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. 2017. Synthesizing Robust Adversarial Examples. *arXiv preprint arXiv:1707.07397* (2017).

[4] Thomas Boutell. 1997. *PNG (Portable Network Graphics) Specification Version 1.0.* Technical Report.

[5] Nicholas Carlini and David Wagner. 2017. Adversarial Examples are not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM.

[6] Lukas Cavigelli, Pascal Hager, and Luca Benini. 2017. CAS-CNN: A Deep Convolutional Neural Network for Image Compression Artifact Suppression. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE.

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3213–3223.

[8] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Binaryconnect: Training Deep Neural Networks with Binary Weights during Propagations. In *Advances in neural information processing systems*.

[9] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to + 1 or -1. *arXiv preprint arXiv:1602.02830* (2016).

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[11] Hugh S Fairman, Michael H Brill, and Henry Hemmendinger. 1997. How the CIE 1931 Color-Matching Functions were Derived from Wright-Guild Data. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur* (1997).

[12] Fisher Yu. 2018. Dilated Residual Networks. https://github.com/fyu/drn [Online].

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in neural information processing systems*.

[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572* (2014).

[15] Google. 2019. A new image format for the Web. https://developers.google.com/speed/webp/ [Online].

[16] Google. 2019. Edge TPU. https://cloud.google.com/edge-tpu/ [Online].

[17] Yiwen Guo, Anbang Yao, and Yurong Chen. 2016. Dynamic Network Surgery for Efficient DNNs. In *Advances In Neural Information Processing Systems*. 1379–1387.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[19] Hewlett Packard Enterprise. 2017. Delivering Accelerated Video Analytics at the Edge for AI Cities. https://h20195.www2.hpe.com/V2/getpdf.aspx/a00004240enw.pdf [Online].

[20] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized Neural Networks. In *Advances in neural information processing systems*.

[21] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. *The Journal of Machine Learning Research* (2017).

[22] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and< 0.5 MB Model Size. *arXiv preprint arXiv:1602.07360* (2016).

[23] Feng Jiang, Wen Tao, Shaohui Liu, Jie Ren, Xun Guo, and Debin Zhao. 2018. An End-to-End Compression Framework based on Convolutional Neural Networks. *IEEE Transactions on Circuits and Systems for Video Technology* (2018).

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet Classification with Deep Convolutional Neural Networks. In *Advances in neural information processing systems*. 1097–1105.

[25] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial Examples in the Physical World. *arXiv preprint arXiv:1607.02533* (2016).

[26] Fengfu Li, Bo Zhang, and Bin Liu. 2016. Ternary Weight Networks. *arXiv preprint arXiv:1605.04711* (2016).

[27] Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang. 2018. Learning convolutional networks for content-weighted image compression. In *CVPR*. IEEE.

[28] Zachary C Lipton and Subarna Tripathi. 2017. Precise Recovery of Latent Vectors from Generative Adversarial Networks. *arXiv preprint arXiv:1702.04782* (2017).

[29] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2016. Delving into Transferable Adversarial Examples and Black-Box Attacks. *arXiv preprint arXiv:1611.02770* (2016).

[30] Zihao Liu, Tao Liu, Wujie Wen, Lei Jiang, Jie Xu, Yanzhi Wang, and Gang Quan. 2018. DeepN-JPEG: A Deep Neural Network Favorable JPEG-Based Image Compression Framework. In *Proceedings of the 55th Annual Design Automation Conference*. ACM.

[31] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. 2018. Conditional Probability Models for Deep Image Compression. In *CVPR*. IEEE.

[32] Microsoft. 2019. Azure IoT Edge. https://azure.microsoft.com/en-us/services/iot-edge/ [Online].

[33] Nvidia. 2019. NVIDIA JETSON SYSTEMS. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems-dev-kits-modules/ [Online].

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic Differentiation in PyTorch. (2017).

[35] Aaditya Prakash, Nick Moran, Solomon Garber, Antonella DiLillo, and James Storer. 2017. Semantic Perceptual Image Compression Using Deep Convolution Networks. In *DCC*. IEEE.

[36] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet Classification using Binary Convolutional Neural Networks. In *European Conference on Computer Vision*.

[37] ITUR Rec. 1995. BT 601: Studio Encoding Parameters of Digital Television for Standard 4: 3 and Wide-screen 16: 9 Aspect Ratios. *ITU-R Rec. BT* 656 (1995).

[38] Iain E Richardson. 2004. *H. 264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia.* John Wiley & Sons.

[39] SJ Robinson and JT Schmidt. 1984. Fluorescent Penetrant Sensitivity and Removability–What the Eye Can See, a Fluorometer Can Measure. *Mater. Eval.* (1984).

[40] Shibani Santurkar, David Budden, and Nir Shavit. 2018. Generative Compression. In *2018 Picture Coding Symposium (PCS)*. IEEE.

[41] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556* (2014).
[42] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One Pixel Attack for Fooling Deep Neural Networks. *IEEE Transactions on Evolutionary Computation* (2019).
[43] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. 2017. Lossy Image Compression with Compressive Autoencoders. *arXiv preprint arXiv:1703.00395* (2017).
[44] Gregory K Wallace. 1992. The JPEG Still Picture Compression Standard. *IEEE transactions on consumer electronics* (1992).
[45] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE transactions on image processing* (2004).
[46] Carole-Jean Wu, David Brooks, Kevin Chen, Douglas Chen, Sy Choudhury, et al. 2019. Machine Learning at Facebook: Understanding Inference at the Edge. In *IEEE HPCA*.
[47] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. 2018. Video Compression Through Image Interpolation. In *ECCV*.
[48] Fisher Yu and Vladlen Koltun. 2016. Multi-scale Context Aggregation by Dilated Convolutions. In *ICLR*.
[49] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. 2017. Dilated Residual Networks. In *CVPR*. IEEE.
[50] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. 2016. Trained ternary quantization. *arXiv preprint arXiv:1612.01064* (2016).

A PROOFS

A.1 Proof of Theorem 1

Proof. Let $G(n,b)$ denote the minimum file size given total n frequency bins and quota c for loss increase, we have:

$$G(n,b) = \begin{cases} (b/n)^n & \text{if } \theta_i \geq b/n, \forall i \in [1,n] \quad (8a) \\ \max\limits_{d_n} d_n G(n-1, b-d_n) & \text{otherwise} \quad (8b) \end{cases}$$

As shown in Fig. 8, we repeatedly apply the above equation to problem (5a) from $n=N$ until the condition in (8a) is met at $n=K$ where

$$d_K < \frac{B-\sum_{i=K+1}^{N} d_i}{K}, \quad d_{K-1} \geq \frac{B-\sum_{i=K}^{N} d_i}{K-1} \tag{9}$$

In Eq. (9), d_i is a variable to optimize, whose value is still undetermined yet. Then the optimal value of problem (5a) can be written as:

$$G(N,B) = \max_{d_N} d_N \ldots \max_{d_K} d_K G(K-1, B-\sum_{i=K}^{N} d_i) \tag{10}$$

From Eq. (9) and AM-GM inequality, we have:

$$G(K-1, B-\sum_{i=K}^{N} d_i) = \Big(\frac{B-\sum_{i=K}^{N} d_i}{K-1}\Big)^{K-1} \tag{11}$$

From Eq. (11) and Eq. (10), we have:

$$\begin{aligned} G(N,B) &= \max_{d_N} d_N \ldots \max_{d_{K+1}} d_{K+1} \max_{d_K} d_K \Big(\frac{B-\sum_{i=K}^{N} d_i}{K-1}\Big)^{K-1} \\ &= \max_{d_N} d_N \ldots \max_{d_{K+1}} d_{K+1} \max_{d_K} F_K(d_K) \end{aligned} \tag{12}$$

To maximize $F_K(d_K)$, we compute $F_K^{'}(d_K)$:

$$F_K^{'}(d_K) = \frac{(B-\sum_{i=K}^{N} d_i)^{K-2}}{(K-1)^{K-1}} \Big(B - \sum_{i=K+1}^{N} d_i - K d_K\Big) \tag{13}$$

From (9) and (13) we have $F_K^{'}(d_K) > 0$. Meanwhile $d_K \leq \theta_K$, thus the optimal solution of $\max_{d_K} F_K(d_K)$ is $d_K = \theta_K$ and:

$$\begin{aligned} G(N,C) &= \theta_K \max_{d_N} d_N \ldots \max_{d_{K+1}} d_{K+1} \Big(\frac{B-\theta_K-\sum_{i=K+1}^{N} d_i}{K-1}\Big)^{K-1} \\ &= \theta_K \max_{d_N} d_N \ldots \max_{d_{K+1}} F_{K+1}(d_{K+1}) \end{aligned} \tag{14}$$

We then keep repeating the steps above from $n=K+1$ to $n=N$, for each iteration, we solve $\max_{d_n} F_n(d_n)$, where

$$F_n(d_n) = d_n \Big(\frac{B-\sum_{j=K}^{n-1}\theta_j - \sum_{i=n}^{N} d_i}{K-1}\Big)^{K-1} \tag{15}$$

With similar method as above, we can prove $F_n^{'}(d_n) > 0, \forall n \in [K,N]$. Given $d_n \leq \theta_n$, the optimal solution is $d_n = \theta_n$. In this way, by going backwards from K to N, we obtain the optimal solution in Theorem 1. □

A.2 Proof of Theorem 2

Proof. From Eq. (1), the conversion from YUV to RGB is:

$$\begin{cases} R &= Y + 2(1-W_R)V \\ G &= Y - 2(1-W_B)\frac{W_B}{W_G}U - 2(1-W_R)\frac{W_R}{W_G}V \\ B &= Y + 2(1-W_B)U \end{cases} \tag{16}$$

According to the chain rule, we have:

$$\begin{cases} \frac{\partial L}{\partial Y} &= \boldsymbol{g}_R \frac{\partial R}{\partial Y} + \boldsymbol{g}_G \frac{\partial G}{\partial Y} + \boldsymbol{g}_B \frac{\partial B}{\partial Y} \\ \frac{\partial L}{\partial U} &= \boldsymbol{g}_R \frac{\partial R}{\partial U} + \boldsymbol{g}_G \frac{\partial G}{\partial U} + \boldsymbol{g}_B \frac{\partial B}{\partial U} \\ \frac{\partial L}{\partial V} &= \boldsymbol{g}_R \frac{\partial R}{\partial V} + \boldsymbol{g}_G \frac{\partial G}{\partial V} + \boldsymbol{g}_B \frac{\partial B}{\partial V} \end{cases} \tag{17}$$

Meanwhile, from Eq. (16), we have

$$\begin{bmatrix} \frac{\partial R}{\partial Y}=1 & \frac{\partial G}{\partial Y}=1 & \frac{\partial B}{\partial Y}=1 \\ \frac{\partial R}{\partial U}=0 & \frac{\partial G}{\partial U}=-2(1-W_B)\frac{W_B}{W_G} & \frac{\partial B}{\partial U}=2(1-W_B) \\ \frac{\partial R}{\partial V}=2(1-W_R) & \frac{\partial G}{\partial V}=-2(1-W_R)\frac{W_R}{W_G} & \frac{\partial B}{\partial V}=0 \end{bmatrix} \tag{18}$$

By plugging Eq. (18) into Eq. (17), we have:

$$\begin{cases} \boldsymbol{g}_Y &= \boldsymbol{g}_R + \boldsymbol{g}_G + \boldsymbol{g}_B \\ \boldsymbol{g}_U &= 2(1-W_B)(\boldsymbol{g}_B - \frac{W_B}{W_G}\boldsymbol{g}_G) \\ \boldsymbol{g}_V &= 2(1-W_R)(\boldsymbol{g}_R - \frac{W_R}{W_G}\boldsymbol{g}_G) \end{cases} \tag{19}$$

To minimize $\overline{\boldsymbol{g}_U} = \sum_{i=1}^{N} \|g_U^i\|/N$ and $\overline{\boldsymbol{g}_V} = \sum_{i=1}^{N} \|g_V^i\|/N$ by finding the optimal W_R, W_G and W_B, following Eq. (19), let $z_1 = W_B/W_G$, and $z_2 = W_R/W_G$, the problems become:

$$\min_{z_1} \sum_{i=1}^{N} \|z_1 - g_B^i/g_G^i\|, \quad \min_{z_2} \sum_{i=1}^{N} \|z_2 - g_R^i/g_G^i\| \tag{20}$$

These are standard least-absolute-deviations (LAD) problems, and the well-known solution is the median:

$$z_1 = \widetilde{\boldsymbol{g}_B/\boldsymbol{g}_G}, \quad z_2 = \widetilde{\boldsymbol{g}_R/\boldsymbol{g}_G} \tag{21}$$

From Eq. (21) and $W_R + W_G + W_B = 1$ from §4.4.1, we get the solution in Theorem 2. □