This ICCV workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Analyzing and Mitigating JPEG Compression Defects in Deep Learning

Max Ehrlich[1] Larry Davis[1] Ser-Nam Lim[2] Abhinav Shrivastava[1]
[1]University of Maryland [2]Facebook AI
{maxehr, lsd}@umiacs.umd.edu sernamlim@fb.com abhinav@cs.umd.edu

Abstract

With the proliferation of deep learning methods, many computer vision problems which were considered academic are now viable in the consumer setting. One drawback of consumer applications is lossy compression, which is necessary from an engineering standpoint to efficiently and cheaply store and transmit user images. Despite this, there has been little study of the effect of compression on deep neural networks and benchmark datasets are often losslessly compressed or compressed at high quality. Here we present a unified study of the effects of JPEG compression on a range of common tasks and datasets. We show that there is a significant penalty on common performance metrics for high compression. We test several methods for mitigating this penalty, including a novel method based on artifact correction which requires no labels to train.

1. Introduction

The JPEG compression algorithm [1] has remained the most popular compression algorithm since the early 90s. Despite rapid advances in video compression and application of those technologies to create superior still image compression algorithms, JPEG is still ubiquitous. JPEG is considered a simple solution to storage and transmission of user data. It is well supported and compresses data quite well despite glaring quality loss at low bitrates. Meanwhile, computer vision, driven by the deep learning revolution of 2012 [2], reaches new milestones regularly with respect to performance, and is continually adopted in the mainstream industry and consumer-facing applications.

Despite this continuing application of deep learning to consumer methods, and the immense popularity of JPEG compression in consumer application, the effect of JPEG compression on deep learning models has been poorly studied. Many consumer applications rely on pretrained models either in whole or in part since labeled datasets are computationally expensive to create. These pretrained models are often designed for academic scenarios and use standard datasets. While many of these datasets, like ImageNet [3], COCO [4], and others, are JPEG compressed, it is often at reasonable quality levels. This is, in general, not reflective of the quality levels seen in consumer applications where the compression level is not always under the control of the deployed system. In light of this, we believe that a comprehensive study which characterizes performance loss due to JPEG compression is long overdue.

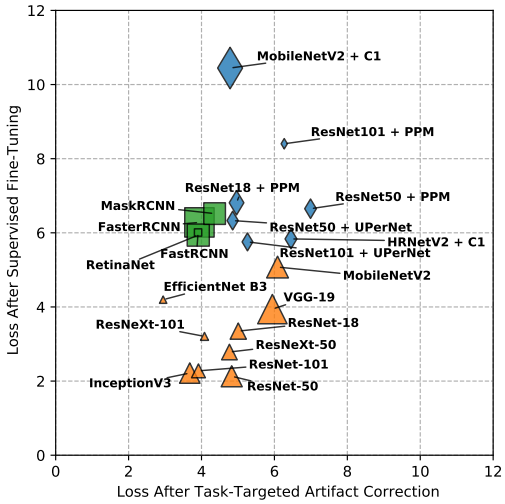


Figure 1: Study results at a glance. Each point shows the performance loss, after applying mitigation, for a model evaluated on quality 10 JPEGs (lower is better), comparing Supervised Fine-Tuning to Task-Targeted Artifact Correction. ▲ Orange triangles are classification models evaluated with Top-1 accuracy on ImageNet, ■ Green squares are detection models evaluated using mAP on MS-COCO, and ♦ blue diamonds are semantic segmentation models evaluated using mIoU on ADE20k. Point **size** shows the inference throughput (larger is better).

In this study, we quantify the effect of JPEG compression over standard performance metrics on a range of datasets and computer vision tasks. We do this by evaluating pretrained models in a systematic and consistent way. In addition to this, we also explore several strategies for mitigating any performance penalty, including a novel method which we call **Task-Targeted Artifact Correction**. This method uses the error of the downstream task logits on JPEG im-

ages with respect to uncompressed versions of the same images, and as such is completely self-supervised: it requires no training labels making it ideal for consumer applications where labels are often expensive to obtain and unreliable. We show that for severe to moderate JPEG compression, there is a steep performance penalty and that this penalty can be mitigated effectively by using our proposed strategies. The study results are summarized in Figure 1, which can be used to quickly choose a model and mitigation technique for a particular application.

Our study helps future scientists make informed decisions when faced with the following two important questions: (1) Is JPEG compression effecting the model? (2) If so, can anything be done about it? Our overall findings are as follows:

1. Heavy to moderate JPEG compression incurs a significant performance penalty on standard metrics, which we show on a set of standard computer vision tasks and datasets. We consider this study to be the primary contribution of our paper and have taken great care to ensure fair comparisons over a plethora of commonly used models. Furthermore, our benchmarking code is fully pluggable and will be released so that models can continue to be evaluated as they are developed. This study is **significantly** more comprehensive than any prior work studying any form of compression (see Section 2 for an overview of prior studies.)
2. If the target application has enough labeled data, **and** preservation of the model result on uncompressed or losslessly compressed images is not important, then fine tuning the task model using JPEG as a data augmentation strategy effectively mitigates this performance loss.
3. If the target application lacks labels, supervised training is impractical, or performance on uncompressed or losslessly compressed images must be preserved, then JPEG artifact correction can be used as a preprocessing step. We show that off-the-shelf artifact correction improves the performance of downstream tasks greatly, and our self-supervised technique makes a further improvement, approaching that of fine tuning the downstream task directly and surpassing it on some tasks all without requiring ground truth labels.

2. Prior Work

We begin by reviewing several recent works which attempt to characterize computer vision task performance under various image corruptions. We then review deep learning methods which make direct use of JPEG data and conclude with a review of artifact correction techniques which consider downstream tasks.

Analysis of Compression Defects. Recently, several works emerged that consider network performance in the presence of JPEG compression. In dos Santos *et al*. [5], the authors revisit Gueguen *et al*. [6] and compare the performance of the previous model under different compression settings. They find that the model still faces a performance penalty for compressed images even though it is trained on DCT coefficients. Mandelli *et al*. [7] use two models from the EfficientNet [8] family and compare multimedia-forensics tasks (camera model identification and generated image detection) with computer vision tasks (ImageNet [3] and LSUN [9] classification). The authors test on several JPEG quality factors as well as examine cropping defects (which misalign the JPEG grid). They find that while all models have a loss of performance on low-quality images, computer vision tasks tended to be more robust than multimedia-forensics tasks. Benz *et al*. [10] show a method for improving batch normalization, which increases robustness to several image corruptions. While they do not treat compression specifically, the ImageNet-C [11] dataset includes compressed images as a type of corruption. Hendrycs and Dietrich, in addition to introducing the ImageNet-C [11] dataset, perform a limited benchmarking of several classification models over several JPEG compression settings, including other corruptions they introduce. An older work examining this problem by Zheng *et al*. [12] considers classification and ranking using a technique called stability training, which tries to match network output on an uncorrupted image with that of the corrupted image. They consider moderate JPEG compression.

Deep Learning with JPEG Data. There have been several works in recent years which attempt to merge deep learning with low-level JPEG primitives. Ghosh and Chellappa [13] include a DCT as part of their initial layer and show a performance improvement on classification. Gueguen *et al*. [6] read JPEG DCT coefficients directly into their network and show that the DCT representation requires fewer parameters to learn comparable results to pixels, yielding a speed improvement. Ehrlich and Davis [14] formulate a fully JPEG domain residual network and again show a speed improvement. Lo and Hang [15] show a method for semantic segmentation on DCT coefficients and show both a performance and speed improvement over using pixels. Deguerre *et al*. [16] show a method for object detection on DCT coefficients and again show performance and speed improvement over pixels. Ehrlich *et al*. [17] formulate a JPEG artifact correction network on DCT coefficients and attain state-of-the-art results for color images. Choi and Han use a network to learn task-guided quantization matrices that maximize task performance after JPEG compression [18].

Artifact Correction for Task Improvement. Several recent works have studied using artifact correction in the presence of downstream tasks [19]–[21]. Galteri *et al*. train a GAN [22] for JPEG artifact correction with the primary goal of making an image for human consumption. In [19],

they use an ensemble of networks for each JPEG quality level and manually pick the network at inference time; in [20] they use an auxiliary network to classify a JPEG to its quality level and automatically pick the artifact correction network at inference time. In both works, they show an improvement on object detection tasks. Katakol *et al*. [21] tests semantic segmentation with several different compression algorithms using adversarial restoration.

3. Methodology

Our goal is to simulate a system receiving a JPEG compressed image compressed at some unknown quality level q[1]. We assume that q is chosen uniformly at random from the range $[10, 90]$, below quality 10 there is little information preserved and above quality 90, the image is nearly identical to the uncompressed version. In all cases, the images are compressed before being put through any transformations that the target model prescribes[2]. For evaluation, the images are compressed at each value in the given range in steps of 10, and each model and each mitigation method are evaluated on these images. We use the Independent JPEG Group's libjpeg [23] software for compression.

For any mitigation methods that require fine tuning, training images are randomly compressed from the same range as a form of data augmentation [24]. The images are always compressed at some quality factor, there are no uncompressed images used for fine tuning. Since the simulated system is assuming JPEG inputs, there is no need for non-JPEG images for fine tuning. The models are trained with a learning rate starting from 1×10^{-3} and ending at 1×10^{-6} for 200 epochs using a cosine annealing [25] learning rate scheduler. We use stochastic gradient descent with momentum for the optimizer with the momentum set to 0.9 and the weight decay set to 5×10^{-4}. As a rule, we only use validation sets for reporting final numbers as most datasets do not provide labeled test data. When training different mitigation techniques for the same model, the same batch size is used [3].

For methods requiring artifact correction, we use QGAC [17]. It is important to understand that correction-based mitigation has only recently become practical as the state-of-the-art has shifted from "quality-aware" models to "quality blinded" models. Quality aware models train a different model for each JPEG quality setting, *i.e*., there would be a unique model for quality 10 JPEGs, quality 20 JPEGs, *etc*. This is a critical limitation because the JPEG quality is not stored in the JFIF file format, making quality-aware models impossible to use in a real setting. Theoretically, these models could be examined in the contrived scenerio of the study, however, this would require a combinatorially large number of models to be trained - one per quality level per AC model per task model. QGAC is quality-blinded meaning that a single model handles all JPEG quality settings. Additionally, it supports color images, achieves state-of-the-art performance on common benchmarks, and has public code and weights making it a prime candidate for use by other researchers. For completeness, we conducted a limited study using two common quality-aware models in Section 4.4 by limiting the compression quality settings and downstream tasks to make the training tractable.

We evaluate the models with no mitigation as a baseline to characterize the effect of JPEG compression when applied directly to machine learning models. We additionally evaluate two common mitigation methods. Finally, we propose a new mitigation method which requires no supervision to train, something which is crucial outside of the academic setting. We now briefly describe these methods.

Baseline. The baseline method simply passes the JPEG compressed evaluation images to the model unchanged. This serves as an important quantifier for how the JPEG quality affects pre-trained models. This method requires no fine tuning.

Supervised Fine-Tuning. In the Supervised Fine-Tuning mitigation, the task network is fine tuned using the protocol given above with JPEG images as input. In other words, we JPEG compress all images from the training set of the given task and fine tune the network from pretrained weights on the given task to improve performance on JPEG inputs. There are two major drawbacks to this method. The first is that it requires a training set of labeled data. This is not always easy to obtain in consumer applications. The second is that it sacrifices performance on uncompressed images in exchange for performance on compressed images. As we show in Section 4, this method provides good results and a fast runtime, especially at training time (See Appendix E for throughput results).

Artifact Correction. For the artifact correction method, we JPEG compress the evaluation images and then perform artifact correction on them before passing them to the task model. This method requires no fine tuning, and we use pretrained weights for the artifact correction model and for the task model. This method does not sacrifice performance on clean images, since these can be detected (by file extension or mime type) and artifact correction can be skipped in this case; however, this method is not trainable and we will show in the results section that it gives worse performance than fine tuning methods.

[1] Most JPEG compression software specifies a scalar quality value in [0, 100] in lieu of a target bitrate, this quality value is not part of the JPEG standard.

[2] For example, cropping to 224×224 for ImageNet based models. Note that while this is more difficult to implement, the reverse process would incur an unrealistically greater loss of quality.

[3] *e.g*., fine tuning a ResNet 18 uses the same batch size as fine tuning artifact correction using resnet18 as the task supervision, but may not use the same batch size as COCO object detection experiments.

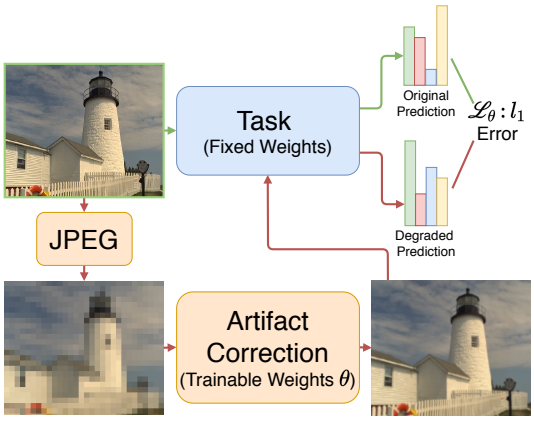


Figure 2: Task-Targeted Artifact Correction. The original image (green) is passed through the task network to obtain the unnormalized prediction logits. Then the image is JPEG compressed and artifact corrected using an artifact correction network with trainable weights. The corrected image is then passed through the task network. The l_1 error between the logits on the original image and the logits on the corrected image is used as loss to tune the artifact correction network weights.

Task-Targeted Artifact Correction. In this novel mitigation technique, we fine tune the artifact correction networking using error on the task network logits. To do this, we minimize the l_1 distance between the task network logits computed on an uncompressed and compressed version of the same image. Formally, given a mini-batch of images B, we minimize

$$\mathcal{L}_\theta = \|\text{Task}(B) - \text{Task}(\text{AC}(\text{JPEG}_q(B); \theta))\|_1, \quad (1)$$

where Task is any task network, AC is the artifact correction network with parameters θ, and q is the JPEG quality level. This is shown schematically in Figure 2. In this way, the artifact correction network learns to correct in a way that maximizes the performance of the downstream network. Note that this method is entirely self-supervised, no ground truth labels are used (the only loss comes from the difference in behavior on the compressed *vs*. uncompressed versions of the images). Not only is this easier to deploy in a consumer setting where labeled data is hard to obtain, but it also alleviates any concern that the artifact correction network may be learning to perform the task for the task network, artificially increasing the number of parameters and making for an unfair comparison. By using l_1 error on the network output, the method is applicable to many different tasks with little or no modification. This method again does not sacrifice performance on uncompressed images since the task network weights are unchanged and an uncompressed input can be detected and the artifact correction step skipped. We show in Section 4 that this method approaches fine tuning the task network directly, even exceeding it in some tasks, despite having no access to ground truth labels.

In addition to this, Task-Targeted Artifact correction supports flexible training scenarios. We show in Section 4.3 that the networks weights are **transferrable** *e.g.*, an artifact correction network which was trained for one network can be used for other networks. This allows a lightweight training setup where a "fast-to-train" network is used to create the artifact correction network weights and then reused for models which would be cumbersome to train. Similarly, Task-Targeted Artifact Correction supports **multi-head** training, where multiple downstream tasks are used at the same time during training. While this method bears a superficial similarity to stability training [12], both transfer and multihead are impossible with stability training and the exact loss formulation we use (Equation 1) is universal: it can be easily applied to many tasks.

4. Results

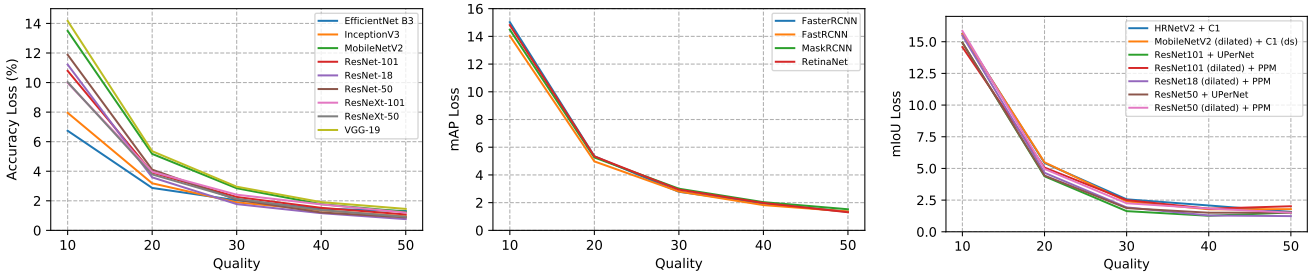
Since our study is focused on both **analyzing** (Sections 4.1 and 4.2) and **mitigating** (Sections 4.3, and 4.4), our results are organized into one of these two categories. First, we show an abridged version of the study on JPEG compression, the full study results are available in Appendix D. After this, we present a preliminary study of two recent forensics models which were tested using the same methodology as the main study. We then examine the transferability and multihead scenarios mentioned in Section 3 in detail, a unique property of Task-Targeted Artifact Correction that makes it a flexible mitigation option.

4.1. Analyzing: Abridged Study Results

We show a subset of our results highlighting interesting, unusual, or edge cases as well as discussing overall findings for each major task. We aim to provide a good covering of common tasks and datasets and picked several models and a single representative dataset per task. While our work is the most comprehensive study of its kind to date, it is not exhaustive: there exist models and datasets which we did not test on. In most cases, we started from pretrained weights available in commodity deep learning libraries.

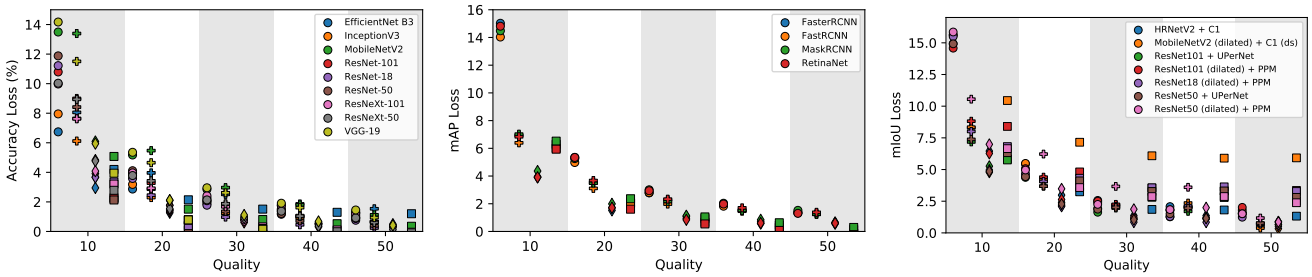
The results presented in the body of the paper are restricted to qualities [10, 50] (heavy to moderate compression) for brevity and because these are the most interesting results. Results on the full range of qualities we considered ([10, 90]) along with tables of results are available in our appendices. In each case, the results are shown as plots giving loss in performance on a task appropriate metric *vs*. JPEG quality level. We studied Classification, Object Detection, Instance and Semantic Segmentation as a set of computer vision tasks.

The abbreviated results are shown in Figure 3, which shows the results of all models with no mitigation separated by task, and Figure 4, which shows the behavior of individual models with mitigations applied compared to no



(a) Classification (b) Detection and Instance Segmentation (c) Semantic segmentation

Figure 3: Performance loss due to JPEG compression by task. The plots show all models from a single task with no mitigation applied. For segmentation tasks, the format of the model name is `Encoder Model` + `Decoder Model` and "ds" indicates that the model was trained with deep supervision. Note that methods which use a Pyramid Pooling Module (PPM) decoder always use deep supervision.



(a) Classification (b) Detection and Instance Segmentation (c) Semantic segmentation

Figure 4: Performance loss of tested models with mitigation applied. ● **Circle**: No Mitigation, + **Cross**: Off-the-Shelf Artifact Correction, ♦ **Diamond**: Task-Targeted Artifact Correction, ■ **Square**: Supervised Fine-Tuning. The models in this figure correspond to those shown in Figure 3. These plots are best viewed digitally and are intended to show the trend of models for each task. For plots with individual models, please see Appendix D.

mitigation in condensed form. Please see Appendix D for full plots and tables for individual models. All performance measures show the absolute drop in performance, *e.g.*, a 14% drop indicates that the model performs 14% worse on JPEG images at the given quality than on uncompressed images. For the reference numbers that we used for this comparison, see Appendix D.4. As a general rule, we observed that more complex tasks incurred a larger performance penalty on JPEG compressed inputs. Additionally, the more complex the task, the more likely it was to be aided by Task-Targeted Artifact Correction, and the less likely it was to be aided by Supervised Fine-Tuning. In Figure 5, we show qualitative results for detection, instance and semantic segmentation. Please see Appendix C to view these results in more detail. We now briefly discuss the details of the study for each task.

Classification. We tested classification models using the ImageNet [3] dataset. We tested the following models: MobileNetV2 [26], ResNet 18, 50, and 101 [27], ResNeXt 50 and 101 [28], VGG 19 [29], InceptionV3 [30], and EfficientNet B3 [8]. The pretrained weights for the task networks in this section come from the torchvision library [31]. The evaluation metric used was Top-1 Accuracy. Models in this task generally responded better to Supervised Fine-Tuning than to Task-Targeted Artifact Correction with the notable exception of MobileNetV2 and EfficientNet which responded better to Task-Targeted Artifact Correction. We ran GradCAM [32] to examine Class Activation Maps for JPEG inputs as well as all mitigations. We found that JPEG degrades the gradient quality as well as induces localization errors. Please see Appendix A for this analysis.

Detection and Instance Segmentation. Next, we show results on object detection and instance segmentation. These models were tested using the MS-COCO dataset [4]. We tested three detection models: Fast R-CNN [33], Faster R-CNN [34], and RetinaNet [35], and we used Mask R-CNN [36] for instance segmentation. The pretrained weights come from the Detectron2 library [37]. In all cases, we use a model with a ResNet 50 [27] backbone, and for the R-CNNs we use a Feature Pyramid Network [38] for the detector. For Task-Targeted Artifact Correction training, we use loss on only the backbone features rather than the detection logits. The evaluation metric used was mean average precision (mAP). Models in this section responded well to all mitigation techniques, with Task-Targeted Artifact Correction helping the most for low-quality settings.

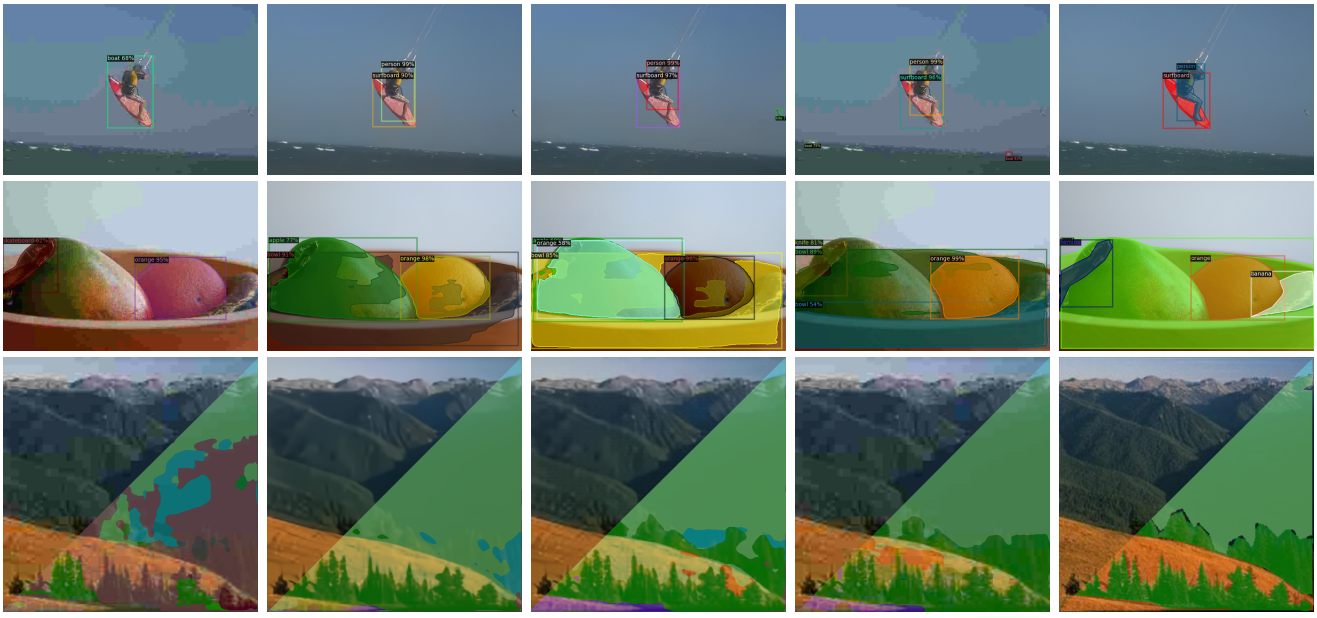


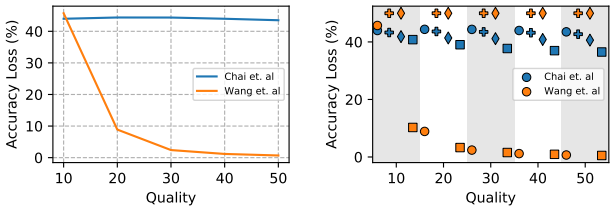
Figure 5: Qualitative results. Top: FasterRCNN Detection, Middle: MaskRCNN Instance Segmentation, Bottom: HRNetV2 + C1 Semantic Segmentation. All inputs were compressed at quality 10. Note the poor quality of the results with no mitigation. In particular, the detection prediction of "boat", while incorrect, is reasonable based on the deformed shape and lack of texture of the compressed image. Despite the compression, the network is displaying some understanding that there is an ocean scene. Similarly, for MaskRCNN, skateboard is predicted seemingly on the shape of the region alone. As mitigations are applied, the confidence of the correct detections for both detection and instance segmentation increases. For semantic segmentation, the result is mostly incorrect with seemingly random classification regions which mitigations are able to clean up significantly.

The moderate quality settings, however, responded better to Supervised Fine-Tuning For a further analysis of detection, we used TIDE [39] and determined that missed detections make up the bulk of errors caused by JPEG compression, please see Appendix B for this analysis.

Semantic Segmentation. For semantic segmentation, we show results on ADE20K using the accompanying code [40], [41]. This code is organized into pluggable encoders and decoders. We test the following encoders: MobileNetV2 [26], ResNet 18, 50, and 101 [27], and HRNet [42]. We test the following decoders: C1 with deep supervision [43], PSPNet (the Pyramid Pooling Module) with and without deep supervision [43], and UPerNet [44]. Similar to the object detection experiments, we train Task-Targeted Artifact Correction using loss from the encoder features only. The metric used is mean of per-class intersection-over-union of classified pixels (mIoU). In general, segmentation models were greatly affected by compressed inputs and Supervised Fine-Tuning was not an effective mitigation, with some models performing worse after fine tuning than with no mitigation at all. Conversely, Task-Targeted Artifact Correction was able to effectively mitigate performance loss for low and moderate qualities.

4.2. Analyzing: Limited Study on Forensics

In addition to the above computer vision tasks, we conducted a limited study on two recent forensics models: Wang *et al*. [24] and Chai *et al*. [45]. The goal of both of these models is to detect an image which, in whole or in part, was generated by a GAN [22], and can be expressed as a two-class classification problem with classes "real" and "fake". Wang *et al*. uses a ResNet 50 architecture [27] to classify the images. Chai *et al*. uses a slightly modified Xception [46] model to classify patches, forming an image label by majority vote among the patches. Both methods use their own bespoke datasets and provide pretrained models which we obtained and used for our experiments. The evaluation metric we used was intended to match the one presented in each paper. For Wang *et al*. that is accuracy of the real/fake prediction, while for Chai *et al*. it is accuracy of the predicted patches. The results are shown in Figure 6. Despite the similar formulation, both networks have completely different behavior on compressed images than traditional ImageNet [3] classification and furthermore have completely different behavior to each other. We note that both models have nearly 100% accuracy on clean images they were trained on. Under JPEG compression, even up to the quality 90 (the maximum that we tested) performance of Chai *et al*. stays around 50% indicating the equivalent to a



(a) Forensic task performance with no mitigation.

(b) Forensics task performance with mitigation

Figure 6: Forensic model results. ● **Circle**: No Mitigation, + **Cross**: Off-the-Shelf Artifact Correction, ♦ **Diamond**: Task-Targeted Artifact Correction, ■ **Square**: Supervised Fine-Tuning.

random guess while Wang *et al*. is able to match its performance by quality 50. While both models incorporated slight JPEG-as-data-augmentation into their pretrained weights, they are greatly aided by additional Fine-Tuning although artifact correction of any kind provided little to no benefit. Since these methods are intended to spot potentially malicious behavior, we find it troubling that a simple perturbation like JPEG is capable of fooling the networks to such a degree and we hope that this study will provide an impetus for further investigation.

4.3. Mitigating: Transferability and Multihead

In this section, we show results which are intended to give a better understanding of our proposed Task-Targeted Artifact Correction. In particular, we examine the transferability of the targeted models, *e.g*., if a model which was trained for one network and task, like a lightweight MobileNetV2 for ImageNet classification, for example, can be used to mitigate performance loss of a more complex and harder to train network like a ResNet-101, or even used in a wholly different task like detection. We also examine if this performance mitigation holds when a correction network is targeted to multiple models simultaneously, in other words, when multiple task networks are providing supervisory signal. These scenarios improve the practical application of the method by allowing for more flexible training and model reuse in addition to the stated advantage of not requiring labels. A table of results for all plots in this section is given in Appendix F.

Intra-Task Transferability. For intra-task transferability, we tested two scenarios. We first trained Task-Targeted Artifact Correction models on ResNet 18 [27] and MobileNet V2 [26], two of our smallest models which are fast to train. We then tested these models on ImageNet [3] classification using ResNet 101 [27] as the downstream network, one of our largest and slowest to train models. The result shown in Figure 7a is comparable with the task-targeted model trained on the ResNet 101 itself, indicating good transfer.

Inter-Task Transferability. For inter-task transferability, we used the same artifact correction models trained in the previous section but now used them to test using COCO [4] object detection on Faster R-CNN [47] and ADE20K [40], [41] semantic segmentation using the HRNetV2 [42] encoder with C1 decoder. These results are shown in Figures 7b and 7c, and again show good transfer with comparable performance to task-targeted networks trained for each of the networks.

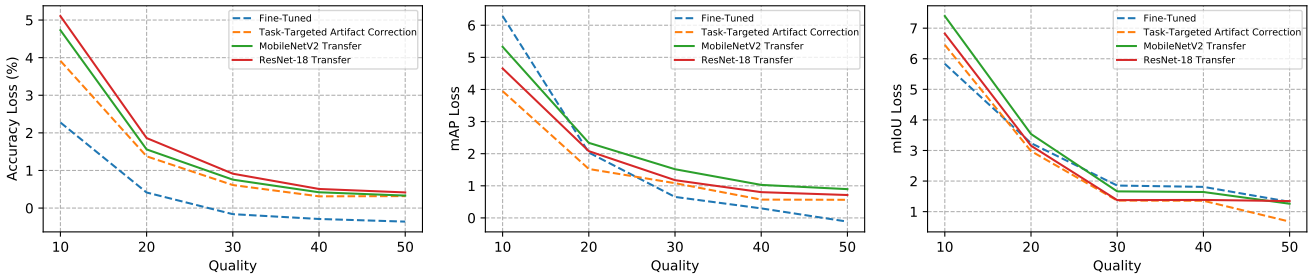
Table 1: Comparison with Common AC Baselines for Task-Targeted Artifact Correction. ARCNN and IDCN are both "quality aware" models. Metrics are classification: top-1 accuracy, detection: mAP, segmentation: mIoU. QGAC outperforms both.

Task (Network)	AC Network	Quality 10	Quality 20
MobileNetV2 (Classification)	ARCNN	61.19	66.89
	IDCN	62.62	65.72
	QGAC	**64.64**	**68.63**
FasterRCNN (Detection)	ARCNN	24.99	27.02
	IDCN	25.99	28.23
	QGAC	**31.43**	**33.85**
HRNetV2 + C1 (Semantic Segmentation)	ARCNN	29.35	35.90
	IDCN	32.62	37.00
	QGAC	**34.14**	**37.61**

Multihead. We tested two setups for multihead training: a two task setup and a three task setup. For the two task setup, we train the artifact correction network using a ResNet 18 [27] as well as a Faster R-CNN [47] providing downstream loss. For the three task setup we add in semantic segmentation using HRNetV2 [42] + C1. To train this, we alternate batches, taking one batch from ImageNet, performing a full forward pass using the Resnet 18, then taking a batch from COCO and performing another full forward pass using the Faster R-CNN backbone and in three task case following the same procedure with ADE20K. The l_1 loss of the features for each separate network is taken and the backpropagated loss is summed. The result, shown in Figure 8 is on-par with artifact correction models trained with a single model, and in the classification case, both multi-head models perform better than the single task corrector indicating better generalization.

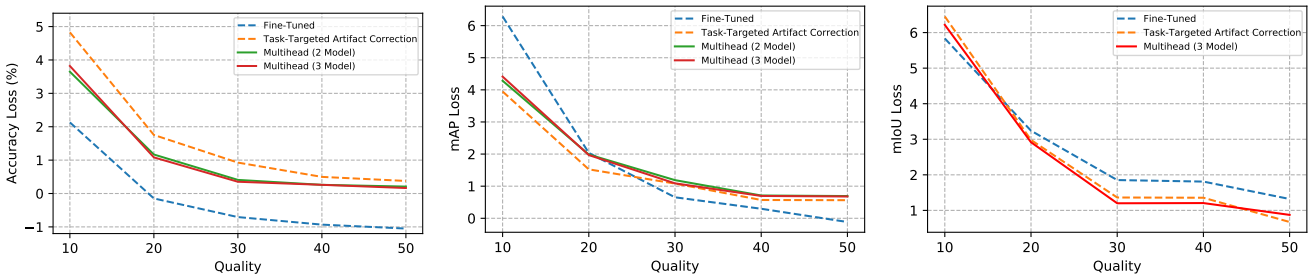
4.4. Mitigating: Limited Comparison with Other Artifact Correction Models

As discussed in Section 3, artifact-correction based mitigation has only recently become viable using quality-blind methods. Using quality-aware methods, which train a unique model for each quality setting, is impossible in real scenarios because the quality setting is not stored in the JPEG file, and in the context of the study, leads to an intractable training protocol which would require models to be trained per-quality level and per-model. Nevertheless, since quality aware models dominated the literature for many years, there is some interest in their behavior in this context. To make this tractable we restrict this part of the study to two models: ARCNN [48], a standard baseline



(a) ResNet101 Results with Transfer. (b) FasterRCNN Results with Transfer. (c) HRNetV2 + C1 Results with Transfer.

Figure 7: Transfer Results. In all plots, we add an evaluation using artifact correction weights that were trained on ResNet-18 and MobileNetV2, our lightest weight models. Note that "Fine-Tuned" and "Task-Targeted Artifact Correction" methods are both trained using their respective task network directly *e.g.* in (a) they use a ResNet 101. **- -** dashed lines indicate results shown in Section 4.1.



(a) ResNet50 Results with Multihead. (b) FasterRCNN Results with Multihead. (c) HRNetV2 + C1 Results with Multihead.

Figure 8: Multihead Results. In all plots, we add an evaluation using artifact correction weights that were trained using multiple task networks. For the two task setup, we used ResNet-50 and FasterRCNN. For the three task setup, we used ResNet-50, FasterRCNN, and HRNetV2 + C1. Note that HRNetV2 + C1 has no two-task multihead model. **- -** dashed lines indicate results shown in Section 4.1.

in artifact correction, and IDCN [49] a recent model. AR-CNN was modified to handle color images and IDCN handles them natively. Both models were ported to our framework and retrained to within 0.5dB PSNR of the published numbers. We additionally restrict the quality settings to 10 and 20 only (the quality settings reported by IDCN). Table 1 shows that both models perform worse than QGAC, further motivating our use of it for the main study.

5. Conclusion

In this paper we conducted a large scale study of JPEG compression on common computer vision tasks and datasets. Our study shows that JPEG compression has a steep penalty across the board for heavy to moderate compression settings. We also tested several strategies for mitigating this performance penalty including a novel method which requires no labels. Our proposed mitigation strategy achieves better results than other unlabeled mitigations and since it requires no labels it is ideal for consumer facing applications where labels are often hard to obtain. For complex tasks, our method outperforms the supervised method despite having no access to ground truth labels. Our method promotes model reuse by allowing transfer of weights between tasks and multihead training. For further results, including throughput for each tested model (Appendix E) we invite readers to our appendices.

We hope to extend this work by considering more compression methods. Despite their relative disuse, there are other still image compression techniques, such as JPEG 2000 [50], HEIF [51], WebP [52], and BPG [53] that should be considered. Consideration of learned-image-compression algorithms is also a useful addition. Finally, video processing models are becoming commonplace, and video compression is almost exclusively lossy. An extended study should consider these models over different video compression settings.

It is our hope that this will be a living study that it will be updated as new models, compression algorithms, and mitigation techniques are developed. Our benchmarking code is pluggable to allow for experimentation with different models, and will be made freely available. Futher, since our Task-Targeted Artifact Correction method shows good transferability, we plan to make the weights we trained for this study available for general use. Many applications will benefit from using our TTAC weights with no modification.

Acknowledgement This project was partially supported by independent grants from Facebook AI, DARPA SemaFor (HR001119S0085) and DARPA SAIL-ON (W911NF2020009) programs.

References

[1] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

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

[4] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[5] S. F. dos Santos, N. Sebe, and J. Almeida, "The good, the bad, and the ugly: Neural networks straight from jpeg," in *2020 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2020, pp. 1896–1900.

[6] L. Gueguen, A. Sergeev, B. Kadlec, R. Liu, and J. Yosinski, "Faster neural networks straight from jpeg," in *Advances in Neural Information Processing Systems*, 2018, pp. 3933–3944.

[7] S. Mandelli, N. Bonettini, P. Bestagini, and S. Tubaro, "Training cnns in presence of jpeg compression: Multimedia forensics vs computer vision," *arXiv preprint arXiv:2009.12088*, 2020.

[8] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[9] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[10] P. Benz, C. Zhang, A. Karjauv, and I. S. Kweon, "Revisiting batch normalization for improving corruption robustness," *arXiv preprint arXiv:2010.03630*, 2020.

[11] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019.

[12] S. Zheng, Y. Song, T. Leung, and I. Goodfellow, "Improving the robustness of deep neural networks via stability training," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 4480–4488.

[13] A. Ghosh and R. Chellappa, "Deep feature extraction in the dct domain," in *Pattern Recognition (ICPR), 2016 23rd International Conference on*, IEEE, 2016, pp. 3536–3541.

[14] M. Ehrlich and L. S. Davis, "Deep residual learning in the jpeg transform domain," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3484–3493.

[15] S.-Y. Lo and H.-M. Hang, "Exploring semantic segmentation on the dct representation," in *Proceedings of the ACM Multimedia Asia on ZZZ*, 2019, pp. 1–6.

[16] B. Deguerre, C. Chatelain, and G. Gasso, "Fast object detection in compressed jpeg images," *arXiv preprint arXiv:1904.08408*, 2019.

[17] M. Ehrlich, L. Davis, S.-N. Lim, and A. Shrivastava, "Quantization guided jpeg artifact correction," *Proceedings of the European Conference on Computer Vision*, 2020.

[18] J. Choi and B. Han, "Task-aware quantization network for jpeg image compression," in *European Conference on Computer Vision*, Springer, 2020, pp. 309–324.

[19] L. Galteri, L. Seidenari, M. Bertini, and A. Del Bimbo, "Deep generative adversarial compression artifact removal," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4826–4835.

[20] ——, "Deep universal generative adversarial compression artifact removal," *IEEE Transactions on Multimedia*, 2019.

[21] S. Katakol, B. Elbarashy, L. Herranz, J. van de Weijer, and A. M. Lopez, "Distributed learning and inference with compressed images," *arXiv preprint arXiv:2004.10497*, 2020.

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

[23] I. J. Group. "Libjpeg." (), [Online]. Available: http://libjpeg.sourceforge.net.

[24] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, "Cnn-generated images are surprisingly easy to spot...for now," in *CVPR*, 2020.

[25] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[28] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[31] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," in *Proceedings of the 18th ACM international conference on Multimedia*, 2010, pp. 1485–1488.

[32] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[33] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[34] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[35] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[36] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[37] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, *Detectron2*, https://github.com/facebookresearch/detectron2, 2019.

[38] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[39] D. Bolya, S. Foley, J. Hays, and J. Hoffman, "Tide: A general toolbox for identifying object detection errors," in *ECCV*, 2020.

[40] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *arXiv preprint arXiv:1608.05442*, 2016.

[41] ——, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[42] K. Sun, Y. Zhao, B. Jiang, T. Cheng, B. Xiao, D. Liu, Y. Mu, X. Wang, W. Liu, and J. Wang, "High-resolution representations for labeling pixels and regions," *arXiv preprint arXiv:1904.04514*, 2019.

[43] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[44] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 418–434.

[45] L. Chai, D. Bau, S.-N. Lim, and P. Isola, "What makes fake images detectable? understanding properties that generalize," *arXiv preprint arXiv:2008.10588*, 2020.

[46] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[47] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[48] C. Dong, Y. Deng, C. C. Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 576–584.

[49] B. Zheng, Y. Chen, X. Tian, F. Zhou, and X. Liu, "Implicit dual-domain convolutional network for robust color image compression artifact reduction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 11, pp. 3982–3994, 2019.

[50] M. W. Marcellin, A. Bilgin, M. J. Gormish, and M. P. Boliek, "An overview of jpeg-2000," in *Proceedings of the Conference on Data Compression*, ser. DCC '00, USA: IEEE Computer Society, 2000, p. 523, ISBN: 0769505929.

[51] M. Hannuksela, E. Aksu, V. M. Vadakital, and J. Lainema, "Overview of the high efficiency image file format," in *JCTVC-V0072*, 2015.

[52] A. WebP, *New image format for the web*, 2020.

[53] F. Bellard, *Bpg image format*. [Online]. Available: https://bellard.org/bpg/.