Investigating the Impact of Adversarial Attacks on AI-Based Image Compression Models

Egor Miroshnichenko [1] Alexey Morozov [1] Timur Nabiev [1] Gennady Shutkov [1]

Abstract

In this paper, we study the robustness of neural image compression (NIC) models against adversarial attacks, especially, we focus our study on the Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). We evaluated different NIC architectures, including the Anchor model and the Attention-based model with varying quality settings. Our study examines how adversarial attacks impact the compression-decompression process, leading to performance degradation characterized by noise and artifacts in the decoded image. In addition, we compare the performance of NIC methods with traditional algebraic compression algorithms. We hypothesize that NIC models are particularly vulnerable to adversarial attacks, which can significantly affect their ability to preserve image fidelity after compression.

Github repo: github.com

Presentation slides: docs.google.com

1. Introduction

In the heyday of the Internet, the problem of storing a large amount of information has become acute. Almost everyone has a mobile phone through which people exchange files, photos, videos, and so on. However, the bandwidth of Internet connections is not unlimited, which has led humanity to the problem of file compression. At the moment, analytical compression methods are doing a good job of this, but they have several drawbacks, such as lossy compression which leads to a significant reduction of image quality, computational complexity, and dependence of compression rate from the data content.

In recent years, artificial neural networks have been widely used for their ability to quickly process large amounts of data and find non-linear dependencies in the data. Neural networks are used mainly for classification and regression tasks (detection of objects in photos and videos, financial risk analysis, and many others). A separate niche in the types of artificial neural networks is occupied by generative models that are used to generate new data. Their architecture consists of two key blocks: an encoder (compresses the source data) and a decoder (decompresses the processed data). This architecture is very similar to the process of compression algorithms, so reports have begun to appear in the scientific community about the use of convolutional neural networks in the problem of data compression. And they show quite good results both in compression ratio and image quality metrics. Specifically, for image quality we use Structural Similarity Index (SSIM) (SSI). Artificial methods are similar in quality to current compression algorithms.

However, the main problem with applying convolutional neural networks to data compression tasks is their extreme sensitivity to external noise, which leads to incorrect operation.

1.1. Principles of entropy models

Currently, context-adaptive entropy models for image compression are the most prominent, achieving better results than traditional image compression methods like JPEG, JPEG2000 and coming on par with the relatively recent VVC codec. Entropy models are based on the Rate-Distortion tradeoff: it must balance between expected bits per pixel, minimised via accurate entropy estimation (rate):

$$R(\hat{y}) = \mathbb{E}[-\log_2 p_{\hat{y}}(\hat{y})] \tag{1}$$

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Egor Miroshnichenko, Alexey Morozov, Timur Nabiev, Gennady Shutkov <Egor.Miroshnichenko@skoltech.ru, Alexey.Morozov@skoltech.ru, Timur.Nabiev@skoltech.ru, Gennady.Shutkov@skoltech.ru>.

Final Projects of the Machine Learning 2023 Course, Skoltech, Moscow, Russian Federation, 2023.

and reconstruction error minimised by the decoder (distortion, D), which is formalized in the following loss-function:

$$\mathcal{L} = \mathcal{R}(\hat{y}) + \lambda \mathcal{D}(x, \hat{x}) \tag{2}$$

where λ controls compression aggressiveness, $y = g_a(x;\phi)$ and $\hat{y} = Q(y)$ are the latent image representation before quantization and compressed code respectively; and x and $\hat{x} = g_s(\hat{y};\theta)$ are the raw and reconstructed images respectively.

2. Related work

2.1. Previous insights

Hyperprior model (Ballé et al., 2018) introduces side information $\hat{z}$ via auxiliary autoencoder:

$$z = h_a(y;\phi_h) \tag{3}$$
$$\hat{z} = Q(z) \tag{4}$$
$$p_{\hat{y}|\hat{z}}(\hat{y}|\hat{z}) \leftarrow h_s(\hat{z};\theta_h) \tag{5}$$

where h_a and h_s are the analysis and synthesis transforms respectively; ϕ_h and θ_h are optimized parameters of the autoencoder.

The main trouble of this approach is that single Gaussian unable to model multi-modal distributions.

Joint-Autoregressive-Hyperprior model is the improvement of the previous idea. As denoted in the name of the model, it ombines hyperprior with autoregressive contexts:

- 5×5 masked convolution captures left/top neighbors.
- Conditions current $\hat{y}_i$ on previously decoded elements.

For hyperprior-utilizing models the loss-function is the mentioned Lagrangian multiplier-based rate-distortion tradeoff with added parameter $\hat{z}$:

$$\begin{aligned}\mathcal{L} &= \mathcal{R}(\hat{y}) + \mathcal{R}(\hat{z}) + \lambda \mathcal{D}(x, \hat{x}) \\ &= \mathbb{E}[-\log_2 p_{\hat{y}|\hat{z}}(\hat{y}|\hat{z})] + \mathbb{E}[-\log_2 p_{\hat{z}|\psi}(\hat{z}|\psi)] + \lambda \mathcal{D}(x, \hat{x})\end{aligned} \tag{6}$$

Factorized density model ψ utilized to encode z as follow:

$$p_{\hat{z}|\psi}(\hat{z}|\psi) = \prod_i (p_{z_i|\psi}(\psi) \mathcal{U}(-\frac{1}{2}, \frac{1}{2}))(\hat{z}_i) \tag{7}$$

where z_i denotes the i-th element of z, and i specifies to the position of each element or each signal.
However, this model is still constrained by unimodal Gaussian assumption.

2.2. Discretized Gaussian mixture likelihoods

The recent paper "Learned Image Compression with Discretized Gaussian Mixture Likelihoods and Attention Modules" by (Cheng et al., 2020). introduces a breakthrough technique for parameterizing latent code distributions.

A Gaussian Mixture Model (GMM) represents a probability distribution as a weighted sum of K Gaussian components:

$$p(y) = \sum_{k=1}^{K} w^{(k)} \mathcal{N}(\mu^{(k)}, \sigma^{2(k)}) \tag{8}$$

where:

- $w^{(k)}$: Mixture weight ($\sum w^{(k)} = 1$)
- $\mu^{(k)}$: Mean of k-th Gaussian
- $\sigma^{(k)}$: Standard deviation of k-th Gaussian

Here, latent codes $\hat{y}$ are quantized integers, requiring adaptation of continuous GMMs to discrete values.

2.2.1. Discretization

During training, quantization is simulated by adding uniform noise $\mathcal{U}(-\frac{1}{2}, \frac{1}{2})$ to latent codes y:

$$\hat{y} = y + \delta, \delta \sim \mathcal{U}(-\frac{1}{2}, \frac{1}{2}) \tag{9}$$

The entropy model is formulated as follow, then:

$$p_{\hat{y}|\hat{z}}(\hat{y}_i|\hat{z}) = \left(\sum_{k=1}^{K} w_i^{(k)} \mathcal{N}\big(\mu_i^{(k)}, \sigma_i^{2(k)}\big) \cdot \mathcal{U}(-\frac{1}{2}, \frac{1}{2}) \right)(\hat{y}_i) \tag{10}$$

where i specifies the location in feature maps.
For integer $\hat{y}_i$, the probability mass is computed via Gaussian CDF Φ:

$$p(\hat{y}_i) = \sum_{k=1}^{K} w_i^{(k)} \left[\Phi\left(\frac{\hat{y}_i + 0.5 - \mu_i^{(k)}}{\sigma_i^{2(k)}} \right) - \Phi\left(\frac{\hat{y}_i - 0.5 - \mu_i^{(k)}}{\sigma_i^{2(k)}} \right) \right] \tag{11}$$

2.2.2. Key Advantages Over Single Distribution Models

Traditional entropy models e.g., (Ballé et al., 2018) use single Gaussian distributions, which fail to capture skewed distributions, which are common in textured regions, heavy tails, which occurs with high-frequency details and multiple modes from repetitive patterns. GMM Advantages in these issues are the following:

- Each mixture component adapts to local statistics;
- Weighting mechanism $w^{(k)}$ automatically prioritizes dominant modes;
- GMM assigns higher precision (lower σ) to smooth regions;
- Preserves bits for complex edges/textures;
- $\hat{y}$ clipped to $[-255, 256]$ during training for stability reasons.
- Edge Cases Handling:
  - For $\hat{y}_i = -255$: Set $\Phi(\hat{y}_i - 0.5) = 0$
  - For $\hat{y}_i = 256$: Set $\Phi(\hat{y}_i + 0.5) = 1$
- Enables 3.8–8.4% bitrate reduction compared to single Gaussian models;

3. Algorithms and Models

In our research we will utilize 2 models (Bégaint et al., 2020) (Kamisli et al., 2024): cheng2020anchor and cheng2020attn. The Cheng2020 models represent two distinct implementations of the Discretized Gaussian Mixture Likelihoods approach. Both models are available in CompressAI model zoo (zoo), offering state-of-the-art performance with architectural variations.

3.1. Cheng-2020 Anchor: Gaussian Mixture Entropy Model

The Cheng-2020 Anchor implements the GMM-based entropy model without integrated attention modules. Its structure follows the baseline architecture from the original paper with four key components:

- Encoder/Decoder Backbone:
  - Encoder: 4× downsampling via strided convolutions. Layer sequence: Conv2D → GDN → Conv2D (stride=2) ×3
  - Decoder: reflects encoder with inverse operations. Layer sequence: Conv2D → IGDN → SubpixelConv2D (upscale=2) ×3
- Hyperprior Network:
  - Auxiliary Encoder: 4x downsampling → Quantization → Factorized Prior
  - Parameter Prediction: outputs 3K channels per spatial position for K=3 mixtures: K weights (softmax-normalized), K means (unbounded), K scales (exp-activated to ensure $\sigma > 0$)
- Training Configuration:
  - Loss: $\mathcal{R} + \lambda \cdot \text{MSE/MS-SSIM}$
  - λ Values: 0.0018 (MS-SSIM) to 0.045 (MSE)
  - Quantization: STE with uniform noise during training

3.2. Cheng-2020 Attn: Attention-Augmented Variant

The cheng2020attn model integrates channel attention modules into both encoder and decoder while retaining the GMM entropy model. Key modifications from the anchor:

- Attention Module Implementation: attention module placed after each GDN/IGDN layer in encoder/decoder and recalibrates channel-wise feature responses, so the modified architecture looks as follow:
  - Encoder: Conv2D → GDN → Attention → Strided Conv2D (stride=2) ×3
  - Decoder: Conv2D → IGDN → Attention → SubpixelConv2D (upscale=2) ×3
- Attention Gradient Effect:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}F} = \text{Attention}(F) + F \cdot \frac{\partial \text{Attention}}{\partial F} \tag{12}$$

  Amplifies gradients for high-entropy regions during backpropagation.
- Convergence: 20% faster than anchor model due to focused feature learning.

3.3. Adversarial Attack Mechanisms

Adversarial attacks can be categorized into several types, including evasion attacks, poisoning attacks, and model extraction attacks. Evasion attacks, which are the focus of this work, involve manipulating input data during deployment to deceive previously trained classifiers. This is in contrast to poisoning attacks, which contaminate the training data to affect model performance during deployment.

We focus on four gradient-based evasion attack methods:

- FGSM (Tamás Muncsan, 2020): Single-step perturbation in gradient direction:

$$x_{adv} = x + \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(x, \theta)) \tag{13}$$

  where x_{adv} is the calculated adversarial image, x is the original image, $\nabla_x \mathcal{L}$ is the loss function, and ϵ is the coefficient controlling the perturbation size.

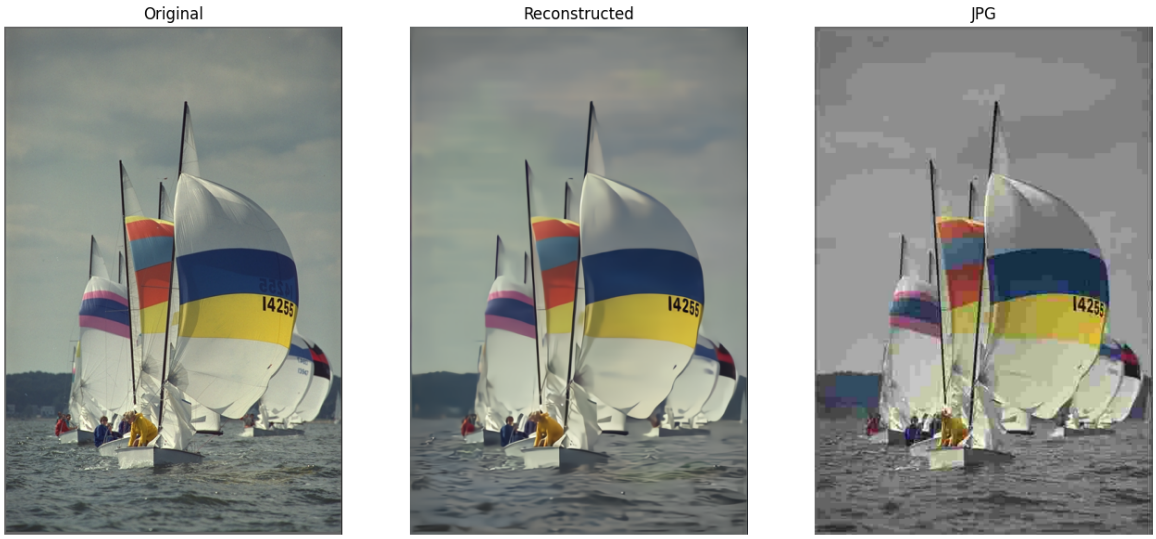

Figure 1. Comparison of AI and JPEG compression. Left image - original, center - AI reconstructed, right - JPEG reconstructed

- I-FGSM (Cihang Xie, 2019): Iterative refinement of FGSM:

$$x^{t+1} = x^t + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(x^t, \theta)) \quad (14)$$

where x^t is the image at iteration t, α is the step size, and the total perturbation is clipped to ensure $||x^{t+1} - x||_\infty \leq \epsilon$.

- M-FGSM (Mohandas S., 2022): Incorporates momentum into I-FGSM:

$$g^{t+1} = \mu \cdot g^t + \nabla_x \mathcal{L}(x^t, \theta) \quad (15)$$

$$x^{t+1} = x^t + \alpha \cdot \text{sign}(g^{t+1}) \quad (16)$$

where μ is the momentum factor and g^t is the momentum term at iteration t.

- PGD (Yudong Chen, 2015): Iterative refinement with projection:

$$x^{t+1} = \text{Proj}_\epsilon \left(x^t + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(x^t, \theta)) \right) \quad (17)$$

where Proj_ϵ projects onto the ϵ-ball around x.

For NIC models, we attack the rate-distortion loss $\mathcal{L} = R(\hat{y}) + \lambda D(x, \hat{x})$, creating perturbations that simultaneously increase bitrate and reconstruction error.

4. Experiments and Results

4.1. Implement AI compression

At the beginning of our study we compared the performance of two compression mechanisms: JPEG and Cheng2020-ancor model. This comparison was performed on the Kodak dataset. We compared the performance of mechanisms with a compression ratio of 1. The results of compression with reconstruction are shown in Figure 1. From that figure, we can see that the neural network copes with compression better than the classical JPEG method. The results of the experiment are presented in the Table 1.

Table 1. Comparisons of compression mechanisms using different metrics

Metric	JPEG	Ch-Anchor	Ch-Attn
PSNR	27.13	30.86	30.74
MS-SSIM	0.89	0.96	0.96
Bit-Rate	0.075	0.071	0.070

Table 1 shows that for all three metrics, the compression algorithm with reconstruction based on the application of artificial neural networks performs better than standard compression algorithms such as JPEG.

Then we compared the performance of two image compression mechanisms for 6 different compression ratios (Figure 2). The following compression ratios were used: 1, 2, 3, 4, 5, 6. We plotted bitrate dependence on two metrics: PSNR and MS-SSIM for two compression mechanisms: JPEG and Cheng2020-Anchor model.

Figure 2 compares the AI compression model (Anchor Variant) and JPEG across the average bit-rates

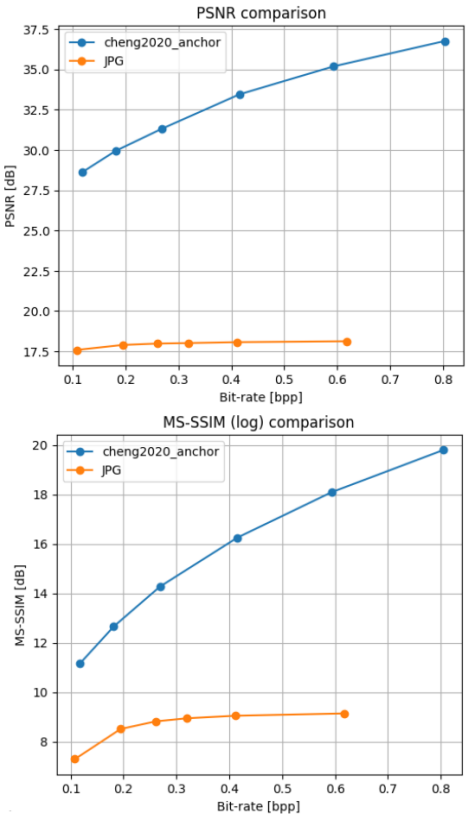

Figure 2. Bit-rate dependence on PSNR and MS-SSIM metrics

(0.1–0.8 bpp), evaluating PSNR (dB) and logarithmic MS-SSIM for the whole dataset. The AI model outperforms JPEG in both metrics at all bit-rates: higher PSNR reflects superior reconstruction fidelity, while MS-SSIM closer to 1 indicates better perceptual quality. Notably, performance gaps widen at lower bit-rates (e.g., 0.1–0.4 bpp), where JPEG's quality declines more sharply. The results highlight the AI model's efficiency in maintaining visual fidelity while reducing bit-rates, particularly under aggressive compression.

4.2. AI compression with attacks

In order to conduct a study on the effect of noise on AI compression, the following attacks were applied to the images: FGSM, I-FGSM, M-FGSM and PGD. Two AI compression models, Cheng-2020 Anchor and Cheng-2020 attn, were analysed for different compression ratios: from 1 to 6.

Algorithm of the experiment:

- Noise generation for the image
- Overlaying noise on an image
- Image compression and reconstruction
- Calculation of PSNR, MS-SSIM and Bit-rate metrics

In Figure 3, you can see the compression and reconstruction results of an image with FGSM and PGD noises using the Cheng2020 Anchor model and I-FGSM and M-FGSM noises with Cheng2020-Attn model for compression ratios 1, 3 and 6.

Graphs of average Bit-rate dependence on PSNR and MS-SSIM for whole dataset for compression ratios from 1 to 6 were plotted (Figure 4). Also for comparison, a curve reflecting the compression of the original (unnoised) image was added to the graph. From these plots, the following conclusions can be drawn:

- Noisy images are recovered noticeably worse than noise-free images.
- Images with I-FGSM, M-FGSM and PGD noise are reconstructed with approximately the same accuracy for all compression ratios.
- The worst reconstructed images are those attacked by FGSM noise.
- The Cheng2020-Attn model dramatically degrades the quality of image reconstruction when the compression ratio is 6.

In addition FGSM attack has the biggest impact on the quality of compression performance since this attack makes several large transforms with large ϵ step size. For example, while the I-FGSM attack does many smaller transformations with smaller steps.

Table 2. Comparison of Attack Methods Using BPP and PSNR Metrics

Attack Method	Quality 1	Quality 3	Quality 6
Bits Per Pixel (BPP)			
PGD	0.77	0.91	1.85
FGSM	0.96	1.09	1.79
I-FGSM	0.765	0.91	1.85
M-FGSM	0.76	0.91	1.85
Peak Signal-to-Noise Ratio (PSNR)			
PGD	29.6	30.0	26.2
FGSM	29.0	28.1	27.3
I-FGSM	30	30.2	26.2
M-FGSM	29.5	29.9	26.3

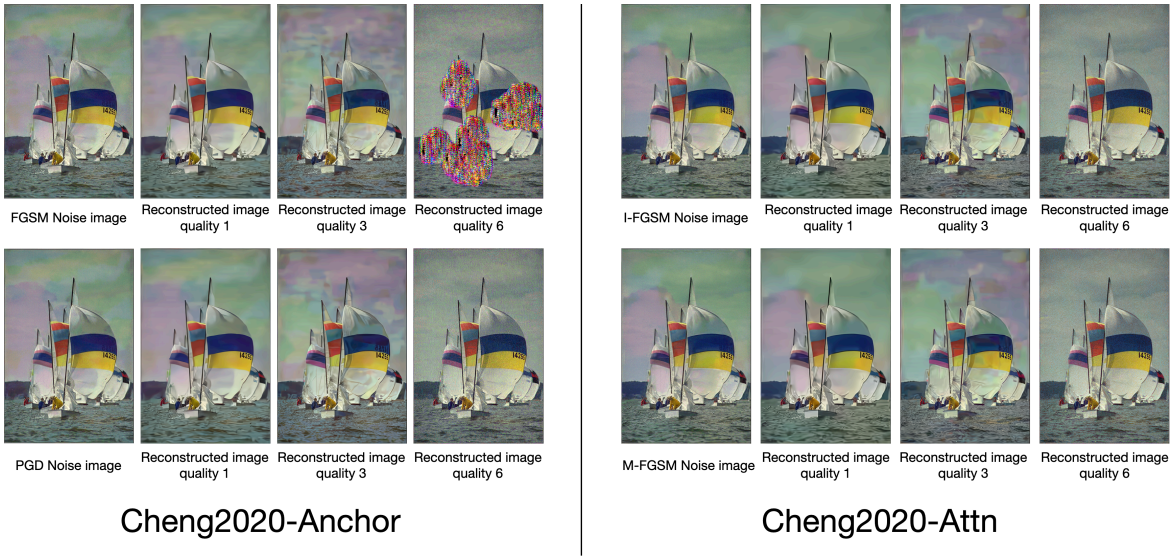

Figure 3. Example performance of compression models (Cheng2020-Anchor on the left, Cheng2020-Attn on the right) for four attack types: FGSM, I-FGSM, M-FGSM and PGD, and three compression ratios: 1, 3 and 6.

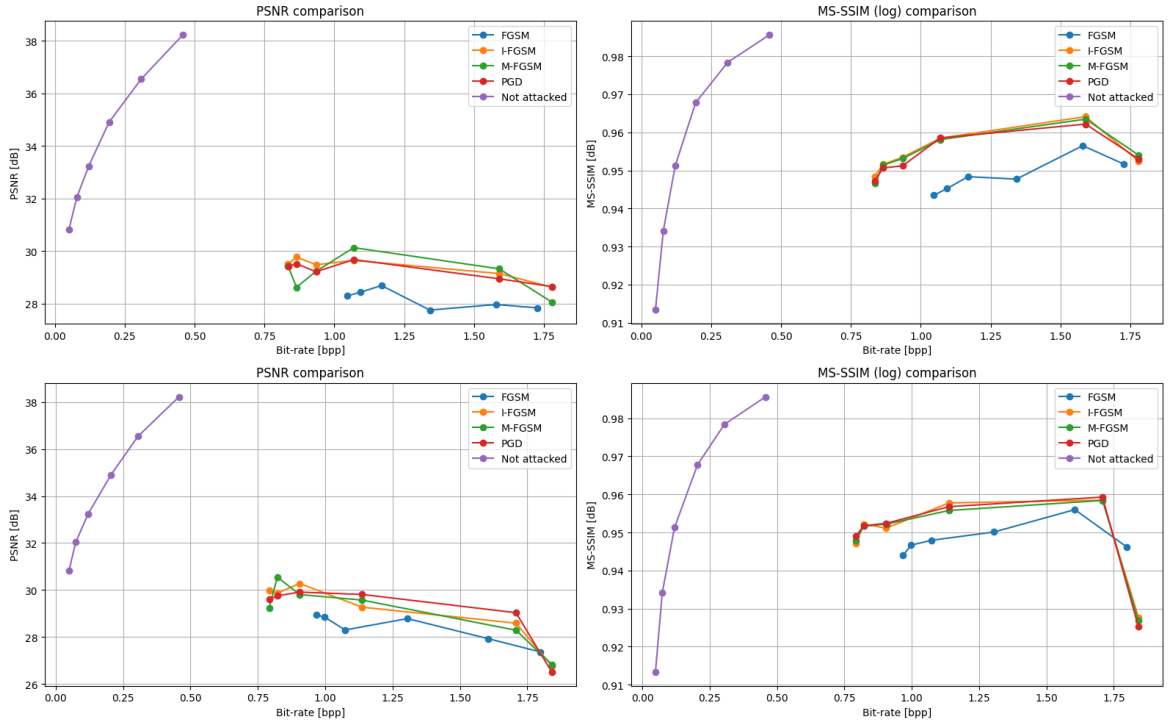

Figure 4. Bit-rate dependence on PSNR and MS-SSIM metrics. Top - Cheng2020-Anchor model, bottom - Cheng2020-Attn model

5. Conclusion

This study investigated the impact of Adversarial Attacks on AI-Based Image Compression Models. Two compression models were tested: Cheng2020 Anchor and Cheng2020 Attn. The study demonstrated the advantage of using neural networks in image compression and reconstruction tasks over conventional compression algorithms such as JPEG using three metrics: Bitrate, MS-SSIM and PSNR.

Both models were also tested on the task of compressing and then reconstructing attacked images with four types of noise: FGSM, I-FGSM, M-FGSM and PGD. Both models have shown their performance in the reconstruction of plague images. However, in some cases with high compression ratio (6 and more), the models could produce artefacts on the reconstructed images, and also the reconstruction accuracy in three key metrics: Bit-rate, PSNR and MS-SSIM decreased significantly.

References

Structural similarity index measure. URL https://en.wikipedia.org/wiki/Structural_similarity_index_measure.

Image compression. URL https://interdigitalinc.github.io/CompressAI/zoo.html.

Ballé, J., Minnen, D., Singh, S., Hwang, S. J., and Johnston, N. Variational image compression with a scale hyperprior. arXiv preprint arXiv:1802.01436, 2018.

Bégaint, J., Racapé, F., Feltman, S., and Pushparaja, A. Compressai: a pytorch library and evaluation platform for end-to-end compression research. arXiv preprint arXiv:2011.03029, 2020.

Cheng, Z., Sun, H., Takeuchi, M., and Katto, J. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 7939–7948, 2020.

Cihang Xie, Zhishuai Zhang, Y. Z. S. B. J. W. Z. R. A. Y. Improving transferability of adversarial examples with input diversity. https://doi.org/10.48550/arXiv.1803.06978, 2019.

Kamisli, F., Racapé, F., and Choi, H. Variable-rate learned image compression with multi-objective optimization and quantization-reconstruction offsets. 2024.

Mohandas S., Manwani N., D. D. P. Momentum iterative gradient sign method outperforms pgd attacks. ICAART (3), pp. 913–916, 2022.

Tamás Muncsan, A. K. Transferability of fast gradient sign method. In Intelligent Systems and Applications, pp. 23–34, 2020.

Yudong Chen, M. J. W. Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. arXiv:1509.03025, 2015.

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Egor Miroshnichenko (25% of work)

- Literature review
- Reporting

Alexey Morozov (25% of work)

- Preparing the GitHub Repo
- Coding main algorithm

Timur Nabiev (25% of work)

- Literature review
- Data collection

Gennady Shutkov (25% of work)

- Coding main algorithm
- Postprocessing

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: We used CompressAI and ImageCompression-Adversarial repositoryes in this study

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: mathematical description of adversarial attacks on images and calculated metrics

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: In the references

4. A complete description of the data collection process, including sample size, is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: in github repo

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: Links in the references

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: in github repo

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   ☐ Yes.
   ☐ No.
   ☑ Not applicable.

   Students' comment:

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: in github repo

9. The exact number of evaluation runs is included.

   ☑ Yes.
   ☐ No.
   ☐ Not applicable.

   Students' comment: in github repo

10. A description of how experiments have been conducted is included.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: in github repo

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    ☑ Yes.
    ☐ No.
    ☐ Not applicable.

    Students' comment: in metrics section

12. Clearly defined error bars are included in the report.

    ☐ Yes.
    ☐ No.
    ☑ Not applicable.

Students' comment:

13. A description of the computing infrastructure used is included in the report.

- [x] Yes.
- [ ] No.
- [ ] Not applicable.

Students' comment: in github repo