Deep Image Compression using
Decoder Side Information
Sharon Ayzik & Shai Avidan

Deep Image Compression using Decoder Side Information

A PRESENTATION BY:

ARIAN TASHAKKOR

Deep-DSIC

Problem Statement

Shifting complexity to decoder-side.

Proposed Method

Using a decoder-side-only compensator combined with an Autoencoder to represent more with less!

Results and Comparisons

with other state-of-the-art methods.

Q&A

Problem Statement

Consider the following use cases:

1. A group of people in an event that upload their pictures of the gathering to an online cloud-based service. These pictures are likely highly correlated although taken at different time steps and with a spatial shift.
2. An image pair from an stereoscopic camera. The image from each camera is stored independently although there is a high degree of correlation between the two.

Problem Statement (Cont’d)

We could make it so that in either use case, the encoder (edge devices in case No. 1 and each of the left and right cameras in case No. 2) uses this correlation in order to compress further compress the results.

Problem:

- Increased load on encoder side while it is more desirable to have the decoder undertake this extra complexity.
- Can we somehow shift this load to the decoder-side only without enforcing extra load on the encoder?

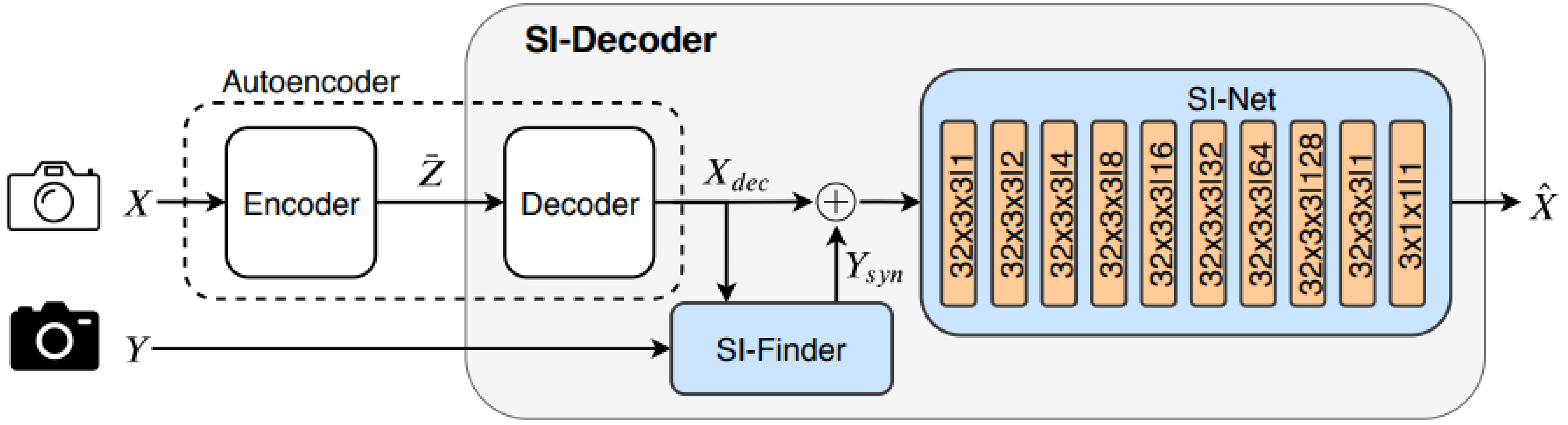
Proposed Method

Distributed Source Coding (DSC) tells us that "it is indeed possible to encode a source X given a correlated source Y even if Y is only available to the decoder side."

General idea:

1. Break both images into patches.
2. Use these patches to measure the correlation between the images.

Proposed Method: Architecture

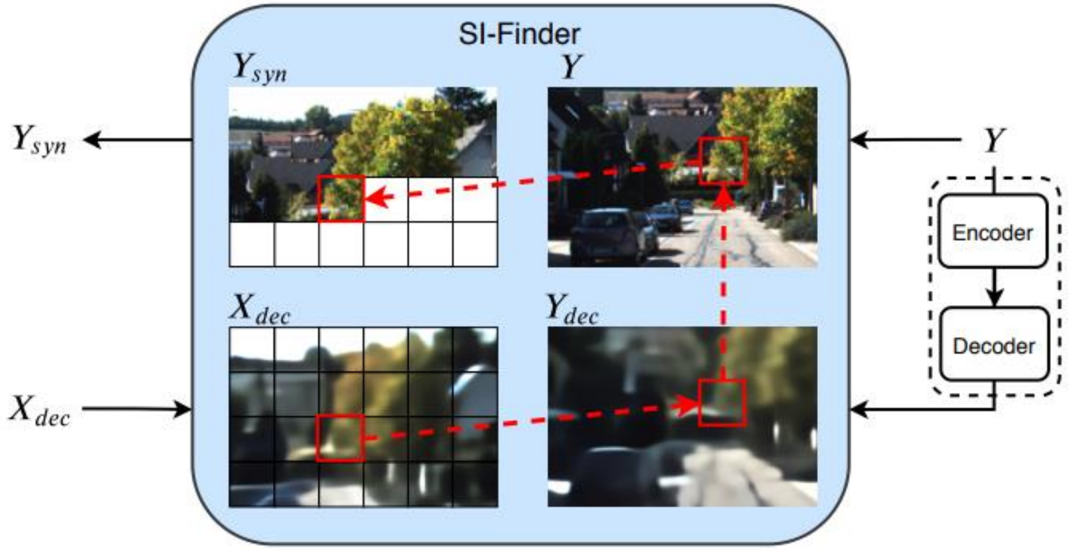


Proposed Method: Breakdown

1. Image X (reference) is encoded to latent representation $\bar{Z}$ and then decoded to X_{dec}.
2. $\mathrm{X}_{\mathrm{dec}}$ along with Y are used to construct Y_{syn}.
3. $\mathrm{X}_{\mathrm{dec}}$ and Y_{syn} are combined and forwarded to the **SI-Net** block that outputs the final reconstruction, $\hat{X}$.

Proposed Method: SI-Finder

- SI-Finder receives X_{dec} and Y.
- Projects Y to the latent space plane, Y_{dec} using the AE.
- Each non-overlapping patch in image X_{dec} is compared to every patch in Y_{dec}.
- Location of the best correlated patch is Y_{dec} is chosen and the corresponding patch is taken from Y. Finally this location is transposed onto Y_{syn} in the corresponding X_{dec} patch location.



Proposed Method: Training & Inference

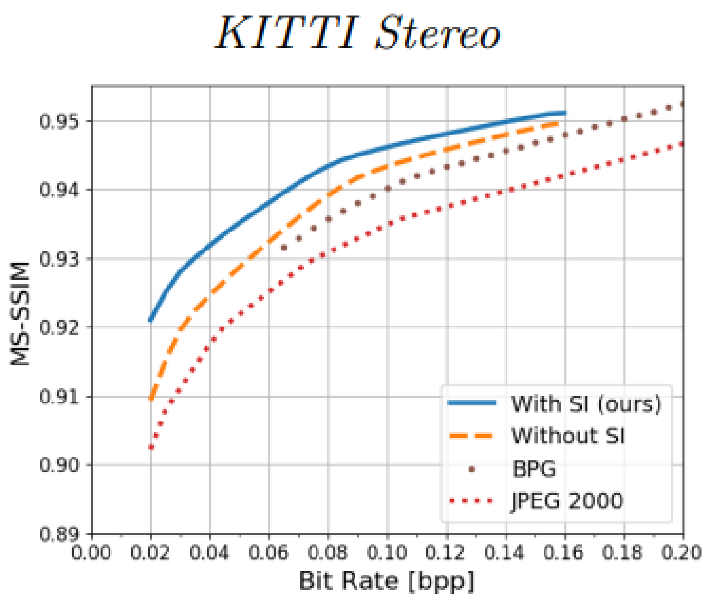
Two sub-networks:

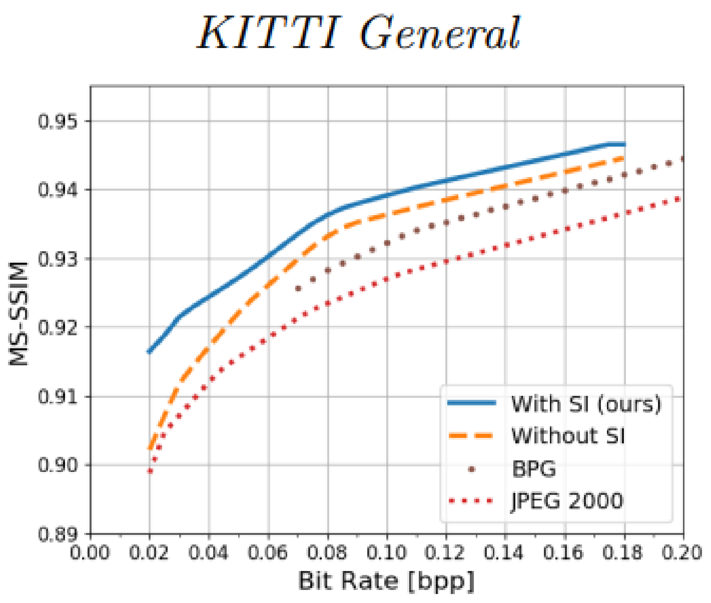
1. An autoencoder designed for image compression: takes input image X and produces decoded image X_{dec}.
2. A secondary network that takes X_{dec} and Y to construct Y_{syn}. Then X_{dec} and Y_{syn} are combined to produce $\bar{X}$.

Both sub-networks **are trained jointly.** At inference time, the encoder only uses the encoder part of the autoencoder, while the decoder uses the rest of the network, **thereby effectively shifting the majority of the complexity to decoder-side where more computational power is at hand.**

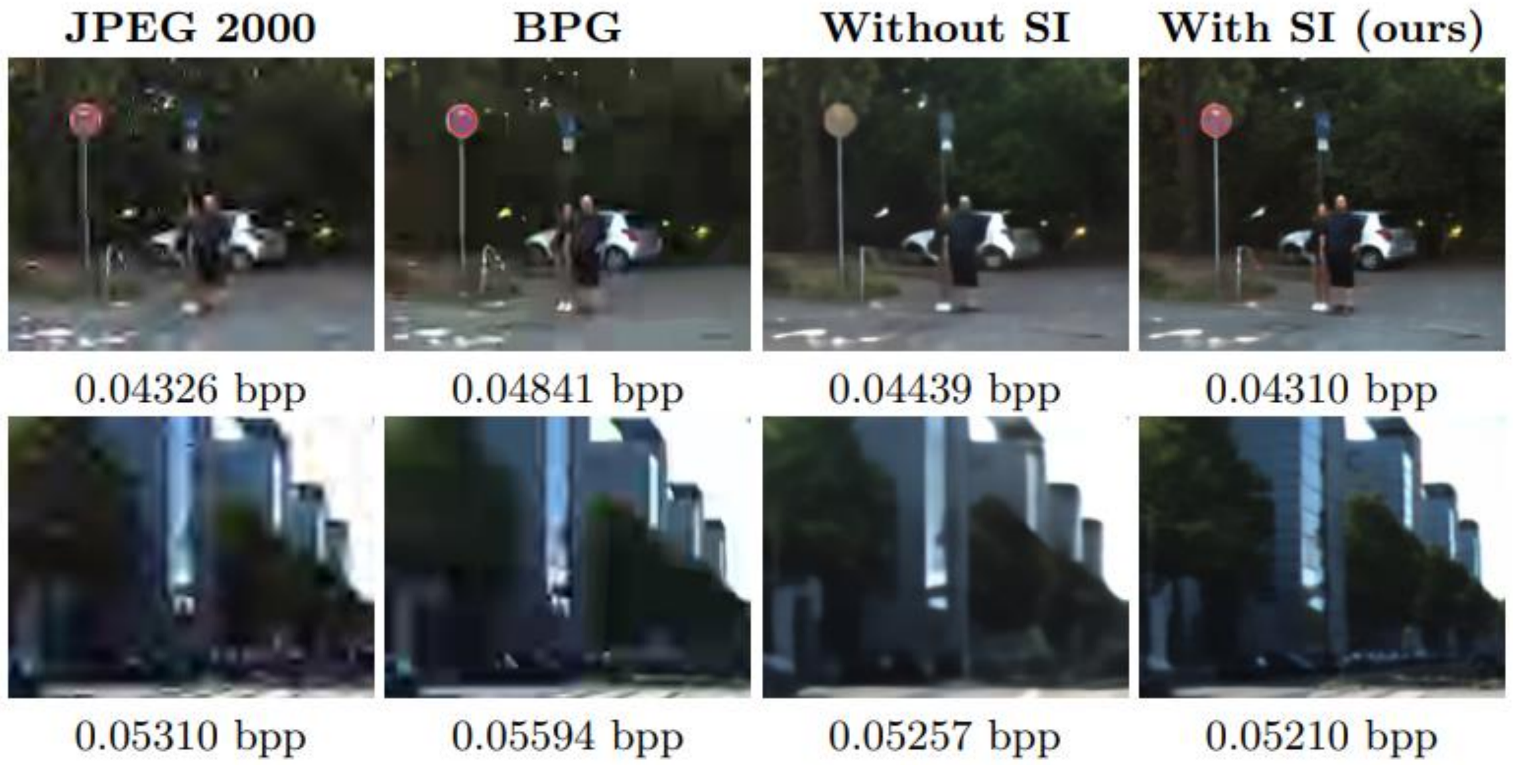
Results and Comparisons

Metric used for comparison is **MS-SSIM** (MultiScale-Structural Similarity Index Measure), a metric that is said to better represent human perception of distortion than MSE and PSNR, especially in cases where distortion is large.



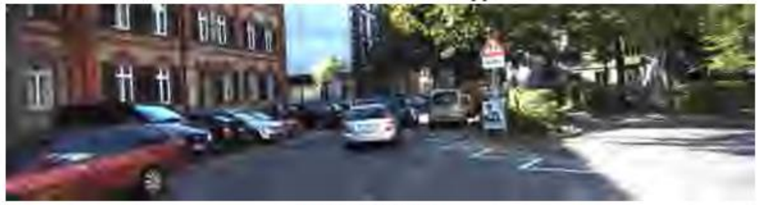


Results and Comparisons (Cont’d)

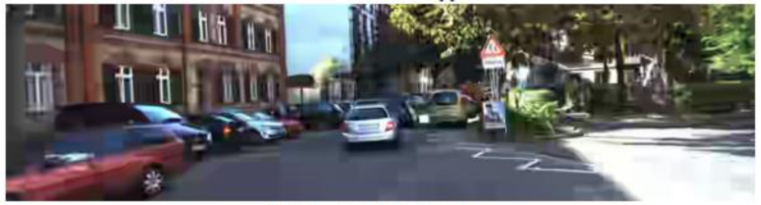


Results and Comparisons (Cont’d)

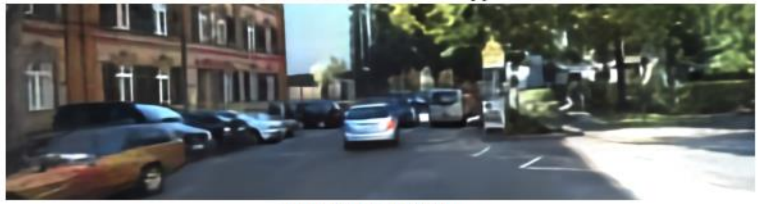
JPEG 2000 0.05225 bpp



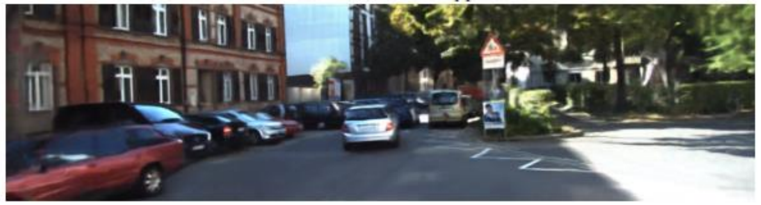
BPG 0.05770 bpp



Without SI 0.05241 bpp

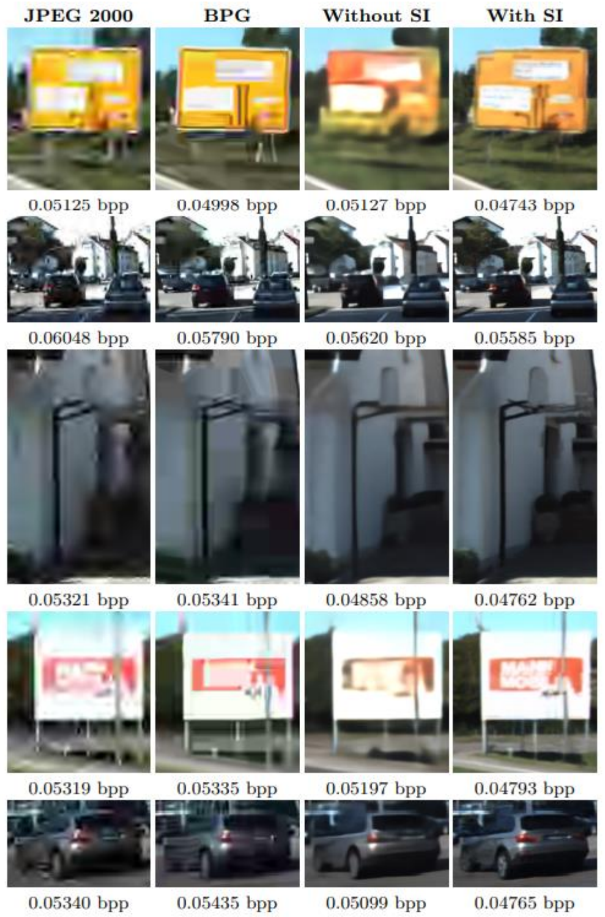


With SI 0.05119 bpp



Results and Comparisons (Cont'd)

The authors' experiments show that "using decoder-only side information can help reduce communication bandwidth by anywhere between **10% and 50%,** depending on distortion level and the correlation between the side information image and the reference image."



Q&A

Thank you for your attention.

Please feel free to ask any questions you might have.

Q: Can this do Lossless Compression?

Short answer: No.

Long answer: according to the authors, “the quantized latent vector $\bar{Z}$ of our auto-encoder is not designed to reconstruct the original image X, nor is it designed to create a coset from which the decoder can [losslessly] recover the correct X. Its goal is only to provide sufficient information to construct a good enough synthetic image Y_{syn} that together with X_{dec} can be used to recover the final result $\hat{X}$ [, an estimation of X]”

Q: Why not apply this to video compression?

Answer: because the side information (which would be the previous frame in the case of a vide) is known both to the encoder as well as the encoder. If side information is already available then we should not purposefully sabotage image quality by not using it in the encoder. Here we consider images that are taken by different cameras at slightly different time steps, which means using side information comes at the cost of bandwidth.

Q: What is DSC?

Answer: [According to Wikipedia] **Distributed source coding (DSC)** is an important problem in information theory and communication. DSC problems regard the compression of multiple correlated information sources that do not communicate with each other. By modeling the correlation between multiple sources at the decoder side together with channel codes, DSC is able to shift the computational complexity from encoder side to decoder side, therefore provide appropriate frameworks for applications with complexity-constrained sender, such as sensor networks and video/multimedia compression (see distributed video coding). One of the main properties of distributed source coding is that the computational burden in encoders is shifted to the joint decoder.

Q: Why not assume the two images are aligned? Match every single pixel instead of patches.

Answer: Because the authors are attempting to solve the problem for the case where the **spatial and temporal shifts eliminate the alignment to a very high extent.** For example, "we would like two images containing a house next to a tree to be correlated even if the tree is to the right of the house in one image, and is to the left in the other."

Q: Loss function of the SI-Net block?

Answer: Reconstruction loss over X_{dec}, i.e.,

$$L\left(X_{dec} \oplus Y_{syn}, \hat{X}\right) = (1 - \alpha).\, d(X, X_{dec}) + \alpha.\, d\left(X, \hat{X}\right) + \beta H(\bar{Z})$$

Where:

$H(.) \coloneqq entropy\ of\ (.)$

$d(.,.) \ \coloneqq distortion\ function$

$\beta \coloneqq scalar\ that\ sets\ the\ trade\ - off\ between\ distortion\ and\ entropy$

$a \coloneqq weight\ of\ the\ final\ system's\ output\ \hat{X}$

Q: How does SI-Finder perform its matching?

Short Answer: the authors provide a method called the “Guided Search” that works under the assumption “that given a patch in X, its corresponding patch in Y should be roughly in the same location in the image plane.”

Long Answer: the authors enforce this assumption using 2D Gaussian mask that helps weight patch similarity in the SI-Finder block. The mean of the mask is taken to be the position of the patch in the image plane, and its variance is roughly half the image size in both axes. This encourages the SI-Finder block to pick patches in Y from roughly the same corresponding location as the patch in X.