Image Compression using Deep Learning

Ruchika Chavhan

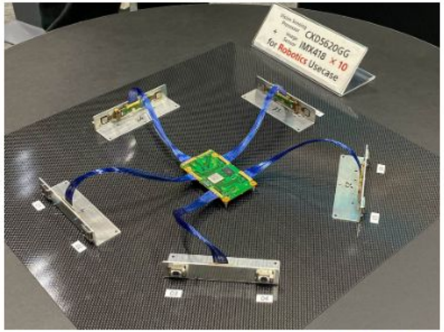
Outline

1. What is Image Compression?
2. Deep Learning Techniques for Image Compression
3. Implementation of models
4. Optimizing results
5. Testing on Jetson Nano
6. Comparing the models
7. Conclusion
8. Future Work

Outline

1. **What is Image Compression?**
2. Deep Learning Techniques for Image Compression
3. Implementation of models
4. Optimizing results
5. Testing on Jetson Nano
6. Comparing the models
7. Conclusion
8. Future Work

Why Image Compression?



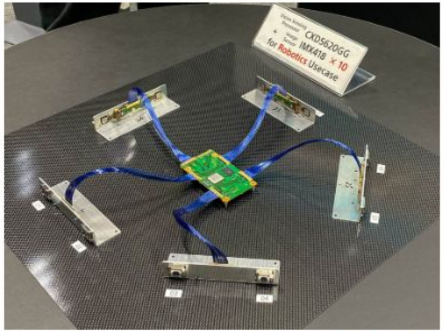
https://www.businessinsider.jp/post-199315

Host PC

- ***Lack of bandwidth, large data cannot be transmitted***
- ***Slow execution of algorithms***
- ***Eradicate redundant information***

Solution: Compressing an image into a lower dimensional vector and restoring it as an image

With image compression



https://www.businessinsider.jp/post-199315

Image compression (on board)

Decompression (on host PC)

How to evaluate performance?

S, ***S'*** are the sizes of image before & after compression

- Compression ratio: S/S'
- Bits per pixel (bpp): S' /total pixels
- PSNR: log inverse of mean squared error
- SSIM: Structural Similarity Index

Conventional methods

- **Image compression**
  - **Loss less**
    - Run length encoding
    - PNG
  - **Lossy**
    - Discrete Cosine Sampling (JPEG)
    - Chroma subsampling

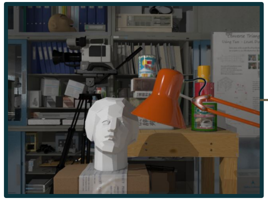
Outline

1. What is Image Compression?
2. **Deep Learning Techniques for Image Compression**
3. Implementation of models
4. Optimizing results
5. Testing on Jetson Nano
6. Comparing the models
7. Conclusion
8. Future Work

Why use Deep Learning Methods?

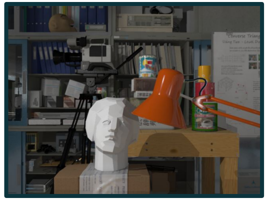
- **More flexible for multi-tasking devices:**
  - Suppose we want to perform classification from input image on host PC
  - Waste of time reconstructing image and then classifying
- Deep Learning methods have recently provided **higher PSNR and SSIM** metrics than JPEG
- Auto encoders are **powerful feature extractors**

Lossy Compressive Auto-encoders



x

Encoder → Quantizer → Decoder



x'

$$\begin{pmatrix} 0.1 \\ 0.2 \\ . \\ . \\ . \\ 0.7 \end{pmatrix} \qquad \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 1 \end{pmatrix}$$

$\mathbf{P_q}$ (empirical prob dist. of the quantization function)

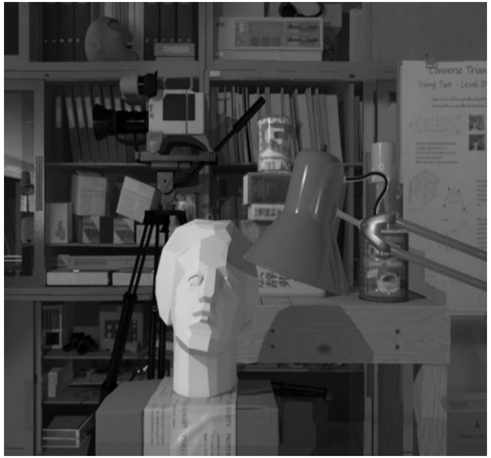
Quantization applied by rounding to the nearest integer, easier to compress in a bitstream

Lossy CAE

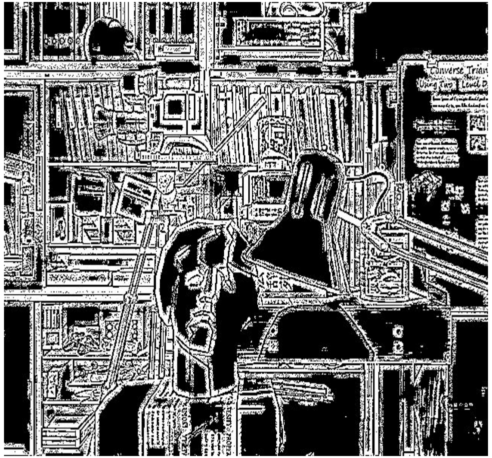
- We need low information Entropy
- Most research is focused on tackling the non-differentiability
  - Stochastic binarization
  - Adding uniform noise to output and using entropy of this dist

$$\text{Minimize } \mathbf{H}[\mathbf{P}_q] + \| x - x` \|_2$$

Using Laplacian as compressed image



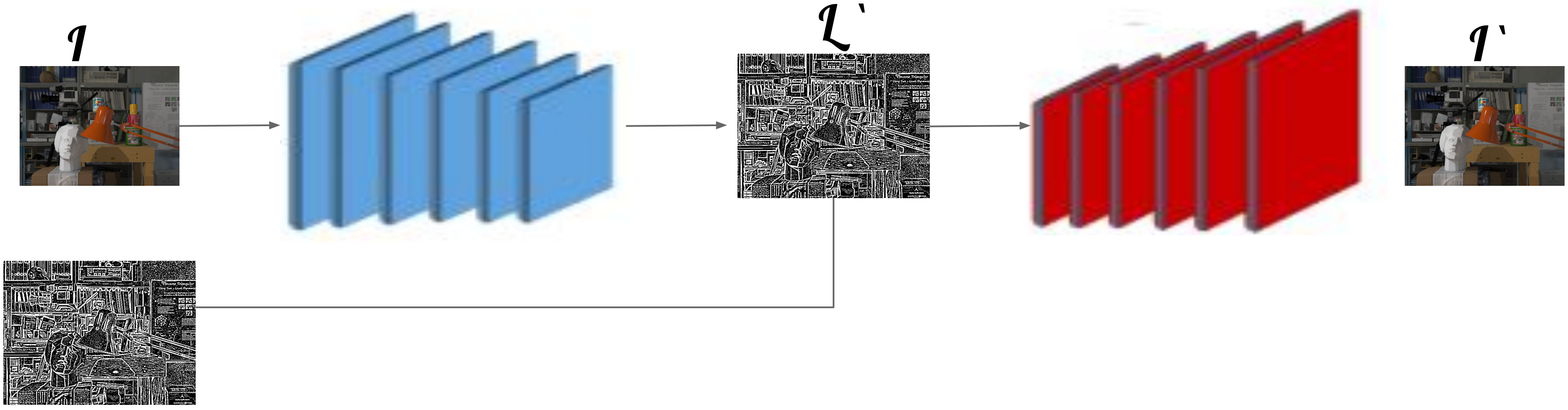
Image



Laplacian

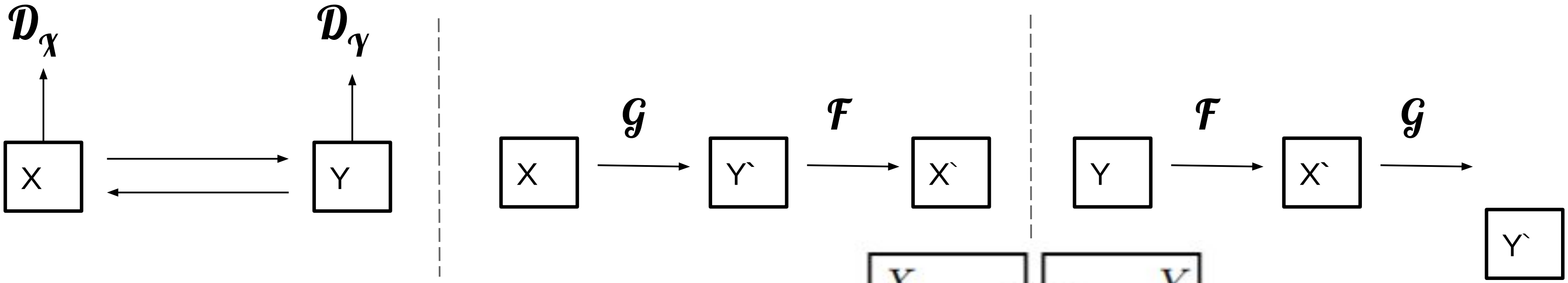
- Most pixels are zero
- Easier to compress
- **Easier to reconstruct**

Auto Encoder: Image to Laplacian

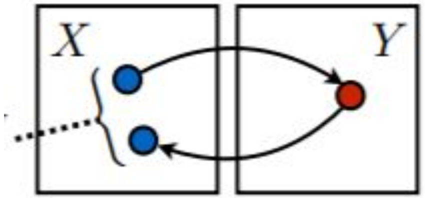


$$\text{Loss} = |\mathcal{I} - \mathcal{I}`| + |\mathcal{L} - \mathcal{L}`|$$

Cycle Consistent GAN: Cycle-GAN



X: image of domain X
Y: image of domain Y
D_X: discriminator that classifies images of domain X
D_Y: discriminator that classifies images of domain Y
G: translates images from domain X to Y
F: translates images from domain Y to X



$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

https://arxiv.org/abs/1703.10593

Cycle Consistent GAN: Optimization

Adversarial loss for G and D_Y

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$

$$\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y).$$

Total Loss

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) =& \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

$$G^*, F^* = \arg \min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y).$$

Outline

1. What is Image Compression?
2. Deep Learning Techniques for Image Compression
3. **Implementation of models**
4. Optimizing results
5. Testing on Jetson Nano
6. Comparing the models
7. Conclusion
8. Future Work

Learning Conditions

- Input image: 1008 x 1008
- Output image: same size as input image
- Compressed image: ½ of input image
- Compression ratio: 4

Environmental conditions

- Training done on Google Colab
  - 15 Gb GPU available
  - 12 Gb RAM available
- We used the Jetson Nano because is it similar to the one used in the real setting
  - GPU: NVIDIA Maxwell architecture with 128 NVIDIA CUDA cores
  - CPU: Quad-core ARM Cortex-A57 MPCore processor
- Host PC:
  - Personal Laptop: HP Pavilion (8GB RAM)

Cycle-GAN models

- Input image: 504 x 504
- Compressed size: 252 x 252
- Compressed image: [0, 255]
- Output image: 504 x 504

Model size	PSNR	SSIM
62 Mb	28.11	0.7511
2 Mb	28.62	0.7942
600 Kb	30.23	0.8035

Original

Reconstructed

- ➢ Image is blurry
- ➢ Structural information is lost
- ➢ Many small details are lost

Original

Reconstructed

- ➢ Slightly more sharper than before
- ➢ Image details and texture is still not reconstructed

Original | **Reconstructed**

- Similar to previously reconstructed images using Cycle-GAN

Auto encoder models

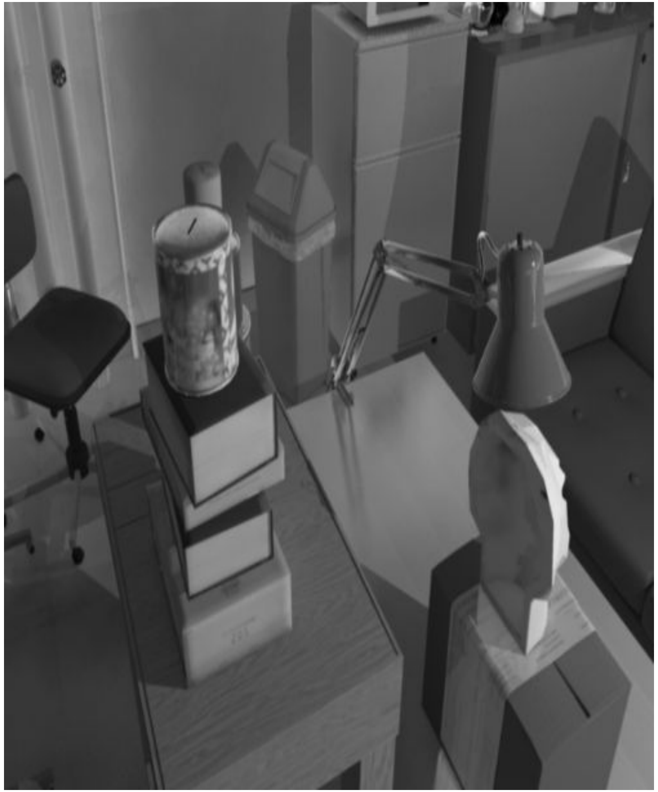
- Input image: 504 x 504
- Compressed size: 252 x 252
- Compressed image: [0, 255]
- Output image: 504 x 504

Model size	PSNR	SSIM
126Mb	27.36	0.7491
4 Mb	25.56	0.6941

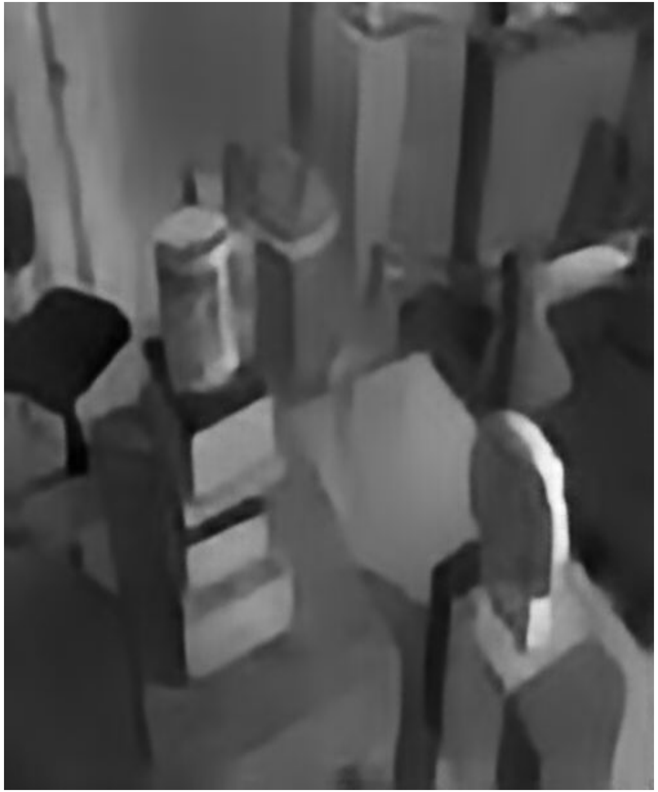
Original	Reconstructed
	

- ➢ Images are very blurry
- ➢ Intensity information is lost
- ➢ Small details are lost

Original



Reconstructed



- Auto encoder models are unable to reconstruct images with minimal loss

Lossy CAE

- Input image: 504 x 504
- Compressed size: n x 126 x 126 (n=8, 32)
- Compressed image: {0, 1}
- Output image: 504 x 504

Model size	PSNR	SSIM
53 Mb	28.88	0.7874
2 Mb	28.56	0.7855

Original

Reconstructed

- Slightly better compared to previous models
- Still, the intensity and structure loss persists

Original

Reconstructed

➢ Similar to previously reconstructed images using Lossy CAE

Outline

1. What is Image Compression?
2. Deep Learning Techniques for Image Compression
3. Implementation of models
4. **Optimizing results**
5. Testing on Jetson Nano
6. Comparing the models
7. Conclusion
8. Future Work

Training the models on larger images

- Problems faced
  - More computations are required
    - Due to increasing in image size
    - This calls for training models of smaller size
    - Smaller models are not good feature extractors
  - Training instability
    - Due to environmental restrictions
    - Google colab offers 15 GB GPU
- We need to train smaller models for larger images without losing structural information.

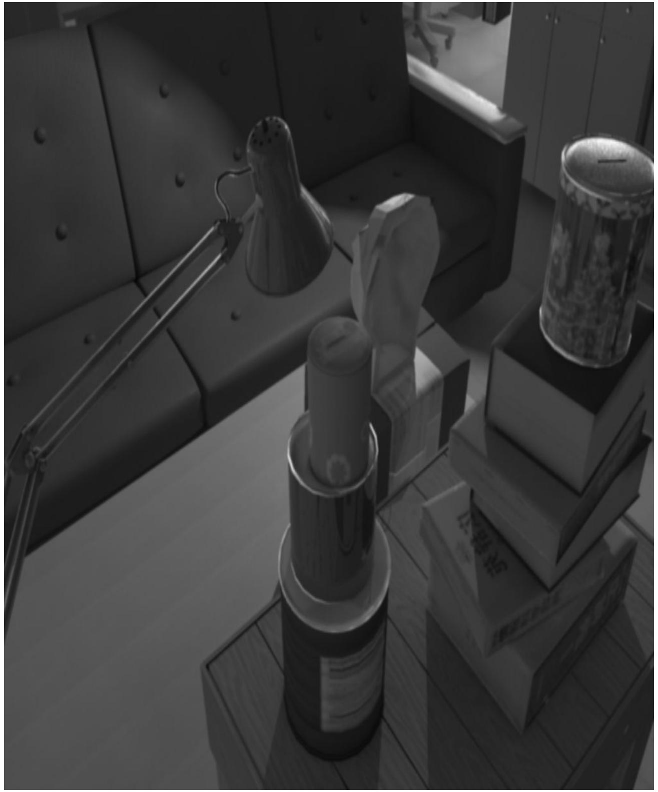
Solution: Preventing information loss

- Better loss function
  - SSIM is a good and robust metric for comparing two images
  - Minimize (1-SSIM) (not many papers/codebases on GitHub have used it!)
- Better performance obtained on smaller models
- Training stability
- Leads to higher PSNR as compared to using MSE

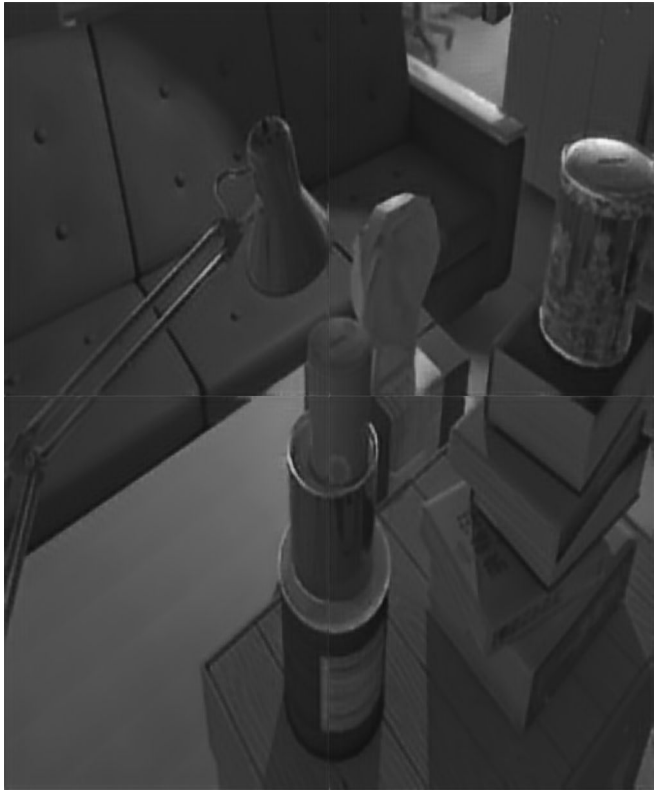
Cycle GAN

- Training involves 4 networks
  - Generator (image to features)
  - Generator (features to image)
  - 2 Discriminators
- GPU limit (15 GB) on Google Colab
- Divide image into patches of 504 x 504 (4 patches)
  - Then pass to the encoder
  - Batch size during testing is 4

Original

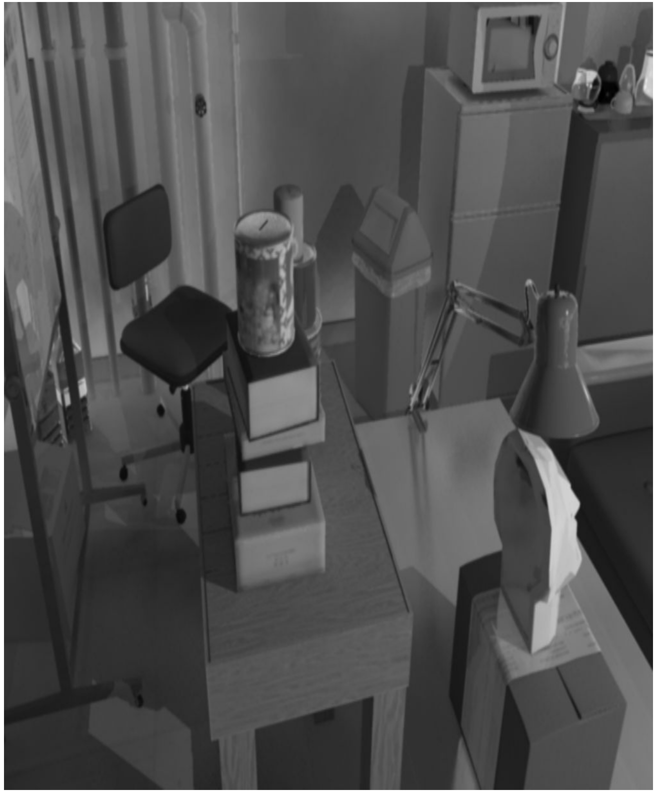


Reconstructed

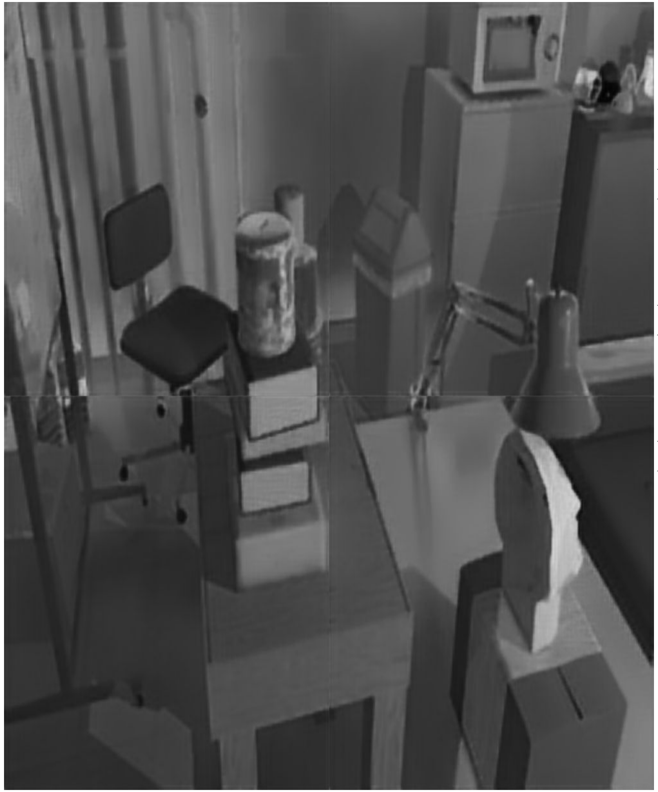


- Compared to previous models, the images are much sharper
- For Cycle-GAN, the lines along which images are patched are still visible

Original



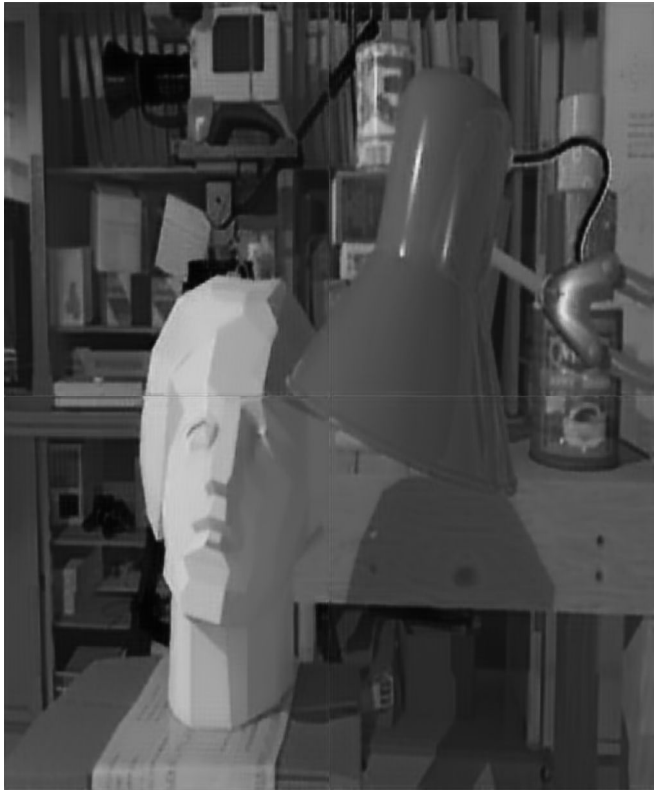
Reconstructed



- ➢ Less intensity loss
- ➢ Decrease in blurriness
- ➢ Structure is maintained

Original

Reconstructed



- The SSIM loss has helped to reconstruct sharper images in case of Cycle GAN

Auto encoder

- Low performance when output of the encoder is constrained to be Laplacian
- Experiments conducted without the Laplacian
- Increase in performance observed
- Input size: 1008 x 1008
- Compressed image: 252 x 252
- **PSNR: 31.17**
- **SSIM: 0.875**

Original

Reconstructed

- ➢ Details of images are preserved while compression

Original

Reconstructed

- ➢ Background objects are also reconstructed with adequate details

Original	Reconstructed
	

- ➢ Slight loss of structure
- ➢ Notice the shelf!

Lossy CAE

- Input size: 1008 x 1008
- Compressed image: 8 x 252 x 252
- **PSNR: 34.01**
- **SSIM: 0.9343**

Original | **Reconstructed**

- ➢ Notice the shelf in this reconstructed image
- ➢ Better results than previous models

Original

Reconstructed

- ➢ Details are well preserved
- ➢ Image is sharper

Original

Reconstructed

- Results of Lossy CAE trained with SSIM as loss outperform previous models

Original

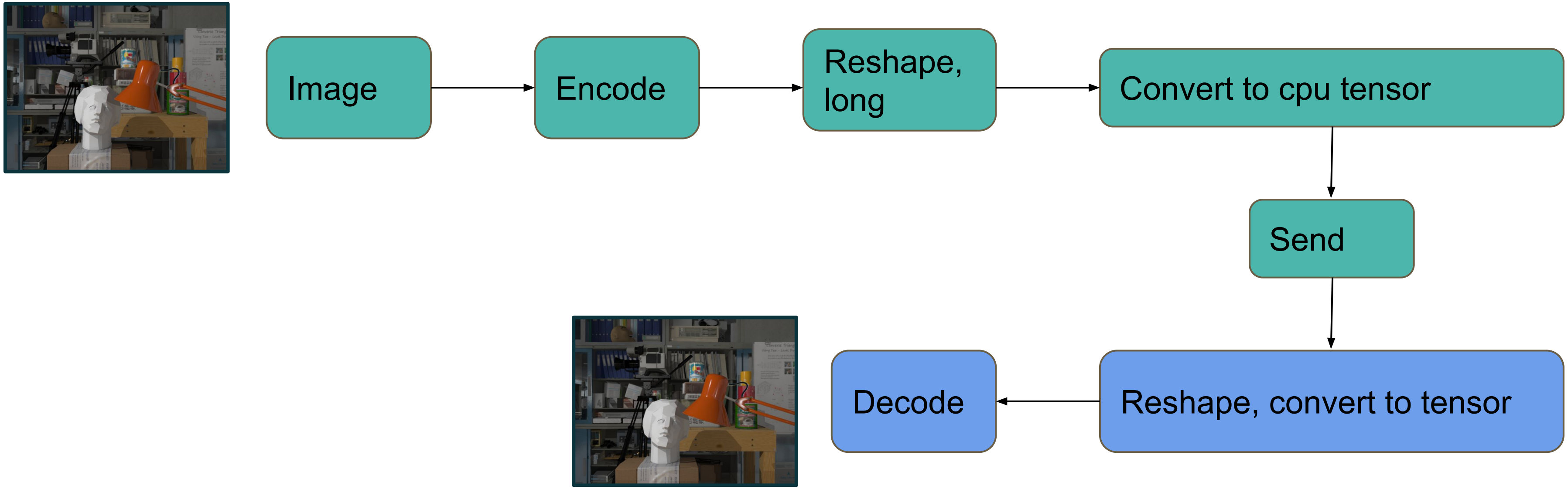
Reconstructed

- ➢ This is a result from lossy CAE trained with MSE loss
- ➢ This results is much more blurry compared to the one in previous slide

Outline

1. What is Image Compression?
2. Deep Learning Techniques for Image Compression
3. Implementation of models
4. Optimizing results
5. **Testing on Jetson Nano**
6. Comparing the models
7. Conclusion
8. Future Work

On the Jetson Nano



What models are considered efficient?

- Basic conditions for a model to be considered good:
  - High PSNR and SSIM
  - Lower model size
  - Average time to encode one image < 33 ms

Cycle GAN

	Model size (PSNR) (Size)	*Encode*	*Misc (encode)*	*GPU -> CPU*	*Pickle*	*Send*	*T_{200}*	*Avg Value*	*Misc (decode) (CPU)*	*Decode (on CPU) (in ms)*
		(in ms)	(in ms)	(ms)	(ms)	(ms)	(sec)	(ms)	(ms)	(in ms)
1	2 Mb (28.62) (504 x 504)	10 - 25	0.2 - 10	0.5 - 1	1 - 4	9 - 15	6.821	**34**	5 - 10	800-1200
2	600 Kb (30.23) (504 x 504)	13 - 25	0.2 - 0.3	0.5 - 1	1 - 4	9- 15	7.084	**33**	5 - 10	400-500
3	600 Kb (30.23) (1008 x 1008)	19 - 25	0.2 - 0.3	0.5 - 1	1 - 4	3 - 4 *(high buffer size)*	7.559	**37**	-----	------

Auto encoder

	Model size (PSNR) (Size)	Encode	Misc (encode)	GPU -> CPU	Pickle	Send	T_{200}	Avg Value	Misc (decode) (CPU)	Decode (on CPU)
		(in ms)	(in ms)	(ms)	(ms)	(ms)	(sec)	(ms)	(ms)	(in ms)
1	126 Mb (27.36) (504 x 504)	30 - 60	0.2 - 1.0	1 - 2	1 - 4	1 - 3	10.259	**51**	-------	--------
2	4 Mb (25.56) (504 x 504)	13 - 65	0.2 - 0.5	0.7 - 3	1- 4	1- 3	8.747	**43**	5 - 10	400 - 700
3	1 Mb (31.71) (1008 x1008)	19 - 28	0.2 - 1	1 - 3	2 - 5	1 - 3	5.971	**30**	------	--------

Lossy CAE

	Model size (PSNR) (Size)	*Encode*	*Misc (encode)*	*GPU -> CPU*	*Pickle*	*Send*	*T_{200}*	*Avg Value*	*Misc (decode) (CPU)*	*Decode (CPU)*
		(in ms)	(in ms)	(ms)	(ms)	(ms)	(sec)	(ms)	(ms)	(in ms)
1	35 Mb (28.88) (504 x 504)	10 - 25	0.2 - 1	3 - 10	3 - 6	100 - 200	8.694	**43**	-----	------
2	2 Mb (28.56) (504 x 504)	7 - 15	0.2 - 1	1 - 3	1 - 3	10-15	4.845	**24**	-----	> 10 seconds
3	2 Mb (34.01) (1008 x 1008)	8 - 15	0.2 - 1	6 - 10	5 - 7	6 - 10 *(high buffer size)*	5.763	**28**	-----	------

Outline

1. What is Image Compression?
2. Deep Learning Techniques for Image Compression
3. Implementation of models
4. Optimizing results
5. Testing on Jetson Nano
6. **Comparing the models**
7. Conclusion
8. Future Work

Comparison

Lowest (for 1008 x 1008)	Model	Size	Best value
Encoding time	Lossy CAE	2 Mb	**8 - 15 ms**
Compressed size	Auto encoder	1 Mb	**252 x 252**
Average time per image	Lossy CAE	2 Mb	**28 ms**
Time to send	All models take almost same time		**3 - 10 ms**

Survey

Model	Compression rate possibility	Processing speed Possibility	Difficulties	Other
AutoEncoder	Can be compressed more (⅛ of original size)	Fast	No constraints on the output of the encoder for better results	(1 - SSIM) loss gives better results
Cycle-GAN	Increasing compression is difficult to train	Slow	Unstable training, Dithering	
Lossy CAE	Can be compressed by a larger size	Fast	Decreasing number of features lead to poor results	Uses stochastic binarization, (1 - SSIM) loss gives better results

Outline

1. What is Image Compression?
2. Deep Learning Techniques for Image Compression
3. Implementation of models
4. Optimizing results
5. Testing on Jetson Nano
6. Comparing the models
7. **Conclusion**
8. Future Work

Conclusions

- From the above experiments, **Lossy CAE** models are the best performing model in terms of image reconstruction quality and processing time required on the Jetson Nano.
- However, the features extracted by encoder of Lossy CAE are larger in size. To obtain features of smaller size, **Auto-encoder** models are the best in terms of size of compressed image.
- SSIM loss has provided a great performance boost for smaller models and can be used in the future to train image reconstruction models

Future work

- Self distillation is a training procedure by which models can be compressed more in size so that the number of computations are lesser. I would definitely try this method to train the Lossy CAE and Auto encoder for performance improvement
- Supposed we want to create a depth map, perform semantic segmentation and surface normal estimation on the host PC. We can use a multi-tasking network that can predict all three with single image by directly sending features. All three tasks will be done on a single feature vector.

Acknowledgements

- I would like to thank Sony Cooperation to give me this great opportunity.
- Thank you to Takuzo Ohara-san, Satoru Mizusawa-san and Koji Ichikawa-san for their valuable guidance
- To Miho Nomura-san and Chinatsu Honda-san for coordinating
- I would further like to thank all members of the TL-13 for a wonderful experience
- My goal in the future is to pursue a PhD in Artificial Intelligence, and this internship has provided me with a great exposure!
- Thank you All !