Video Compression Final Project

江梓豪　　郭家瑋

309551116　309552003

L^2C – Learning to Learn to Compress

Nannan Zou*, Honglei Zhang§, Francesco Cricri§, Hamed R. Tavakoli§,
Jani Lainema§, Miska Hannuksela§, Emre Aksu§, Esa Rahtu*
*Tampere University, Tampere, Finland
{nannan.zou, esa.rahtu}@tuni.fi
§Nokia Technologies, Tampere, Finland
{honglei.1.zhang, francesco.cricri, hamed.rezazadegan_tavakoli, jani.lainema, miska.hannuksela, emre.aksu}@nokia.com

Outline

- **Motivation**
- **Proposed method**
- **Experimental results**

Motivation

Problem

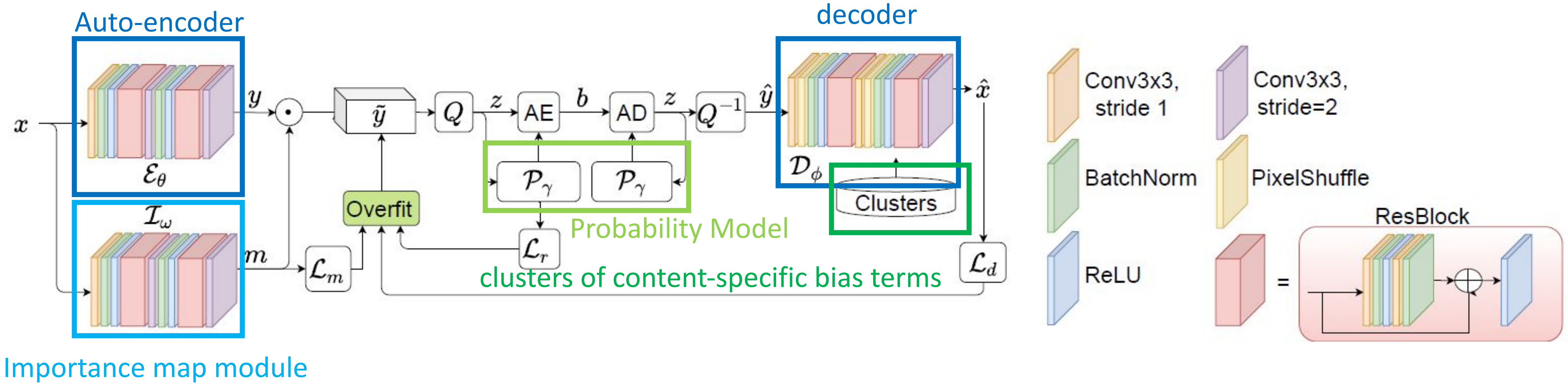
- Current architecture suffer from domain shift — content type at test time is different from the content type considered during training
- The neural networks in the codec are not optimized on every unseen test image

Goal

- Adaption / overfitting to the input latent
- Reduce the gap between training and inference

Proposed Method — L²C

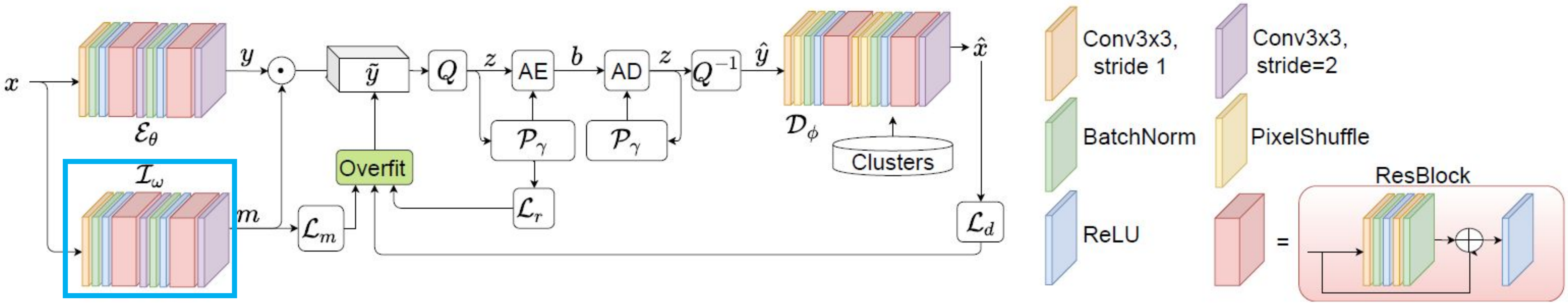
Overview of L²C



- ε_θ: *encoder network parametrized by weight* θ
- $\mathcal{D}_\phi$: *decoder network parametrized by weights* ϕ
- I_ω: *importance map module parametrized by weights* ω

Proposed Method — L^{2}C

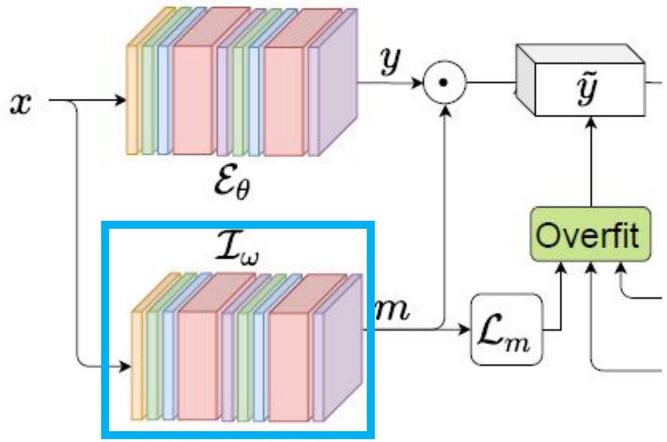
A. Learned Spatially-varying Channel Masking



Importance map module

Proposed Method — L²C

A. Learned Spatially-varying Channel Masking



Importance map module

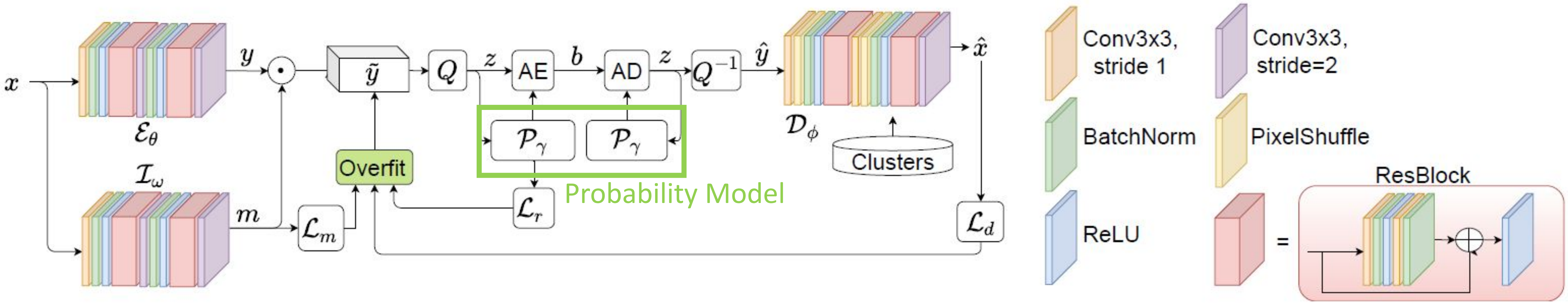
- Allocate a varying number of channels to different spatial areas of the encoded tensor y
- Output and importance map $\tau \in \mathbb{R}^{\frac{H}{s},\frac{W}{s},1}$ with elements in [0, 1]
- This map is then quantized with bits $\log_2 c$ and then expanded into a mask $m \in \mathbb{R}^{\frac{H}{s},\frac{W}{s},c}$:

$$m_{i,j,k} = \begin{cases} 1 & \text{if } k < c\tau_{i,j} \\ 0 & \text{otherwise.} \end{cases}$$

- Constraint: $\mathcal{M}(\tau) = |\bar{\tau} - \zeta|$
- $\bar{\tau}$ is the mean value of τ and ζ is a constant representing the target average non-zero ratio in m

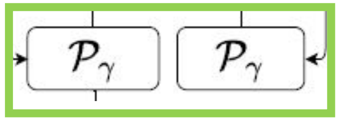
Proposed Method — L^2C

B. Probability Model for Lossless Coding



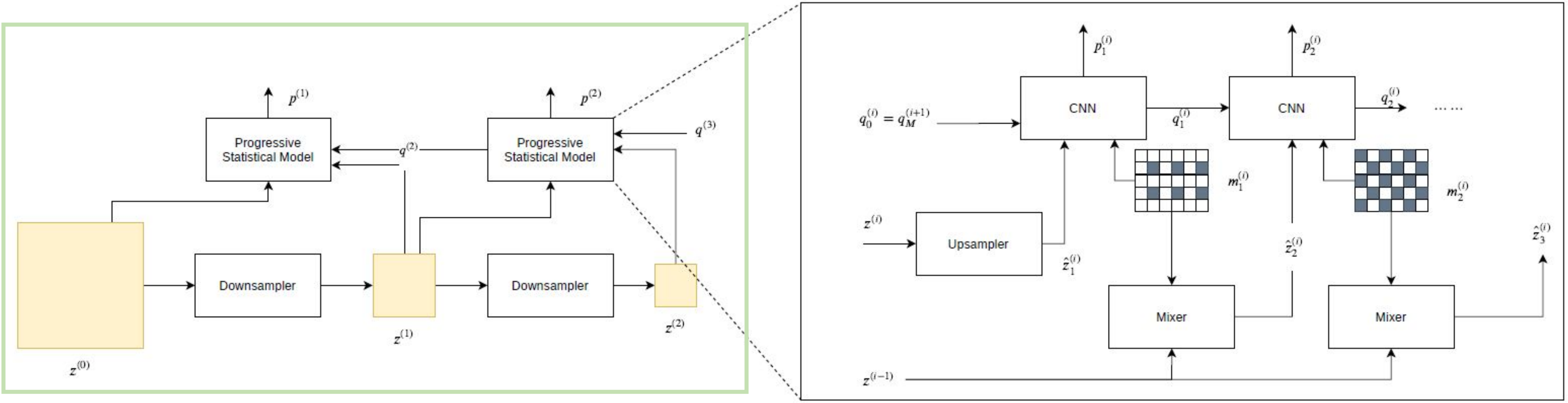
[1] *H. Zhang, F. Cricri, R. Tavakoli, N. Zou, E. Aksu, and M. Hannuksela, "Lossless Image Compression Using a Multi – Scale Progressive Statistical Model,"* ACCV, 2020.

Proposed Method — L²C



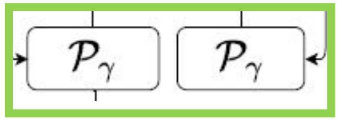
Probability Model

B. Probability Model for Lossless Coding



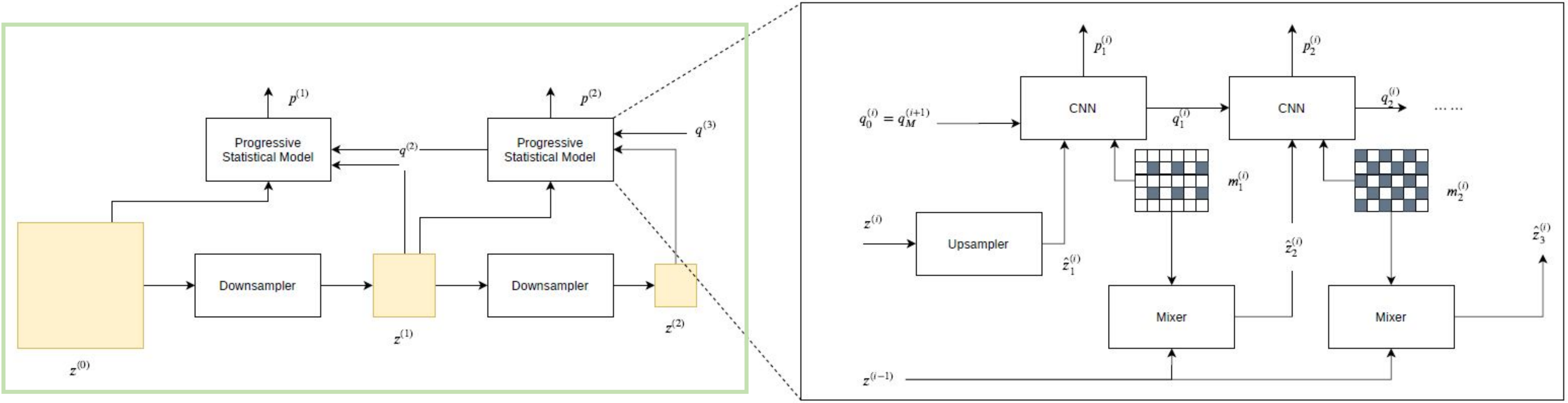
- Downsample an input image into a number of low-resolution representations
- Used as a context to estimate the distribution function of the pixels in a higher resolution representation.

Proposed Method — L^{2}C



Probability Model

B. Probability Model for Lossless Coding

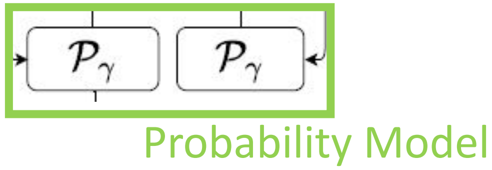


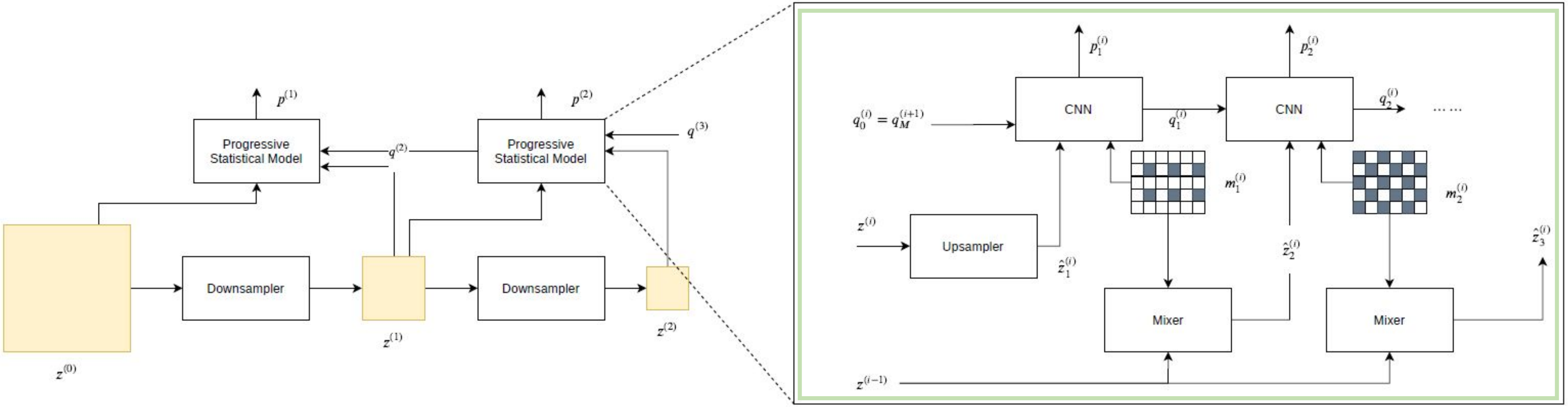
- $z^{(0)}$: *Latent*
- $z^{(i)}$: *low resolution representation of* $z^{(0)}$ *at scale* $i, i = 1,2,\dots,M$
- *The joint distribution function of elements in* $z^{(0)}$ *is defined by* $p(z^{(0)}) = \left(\prod_{i=1}^{M-1} p\left(z^{(i-1)} | z^{(i)}\right)\right) p\left(z^{(M)}\right)$
- $\mathcal{L}_r(z) = -\mathrm{E}(\log(\mathrm{p}(z^{(0)})))$

Proposed Method — L²C

B. Probability Model for Lossless Coding

1	2	1	2	1	2
3	x	3	x	3	x
1	2	1	2	1	2
3	x	3	x	3	x



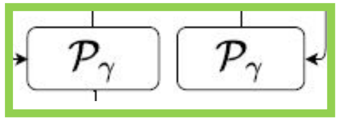


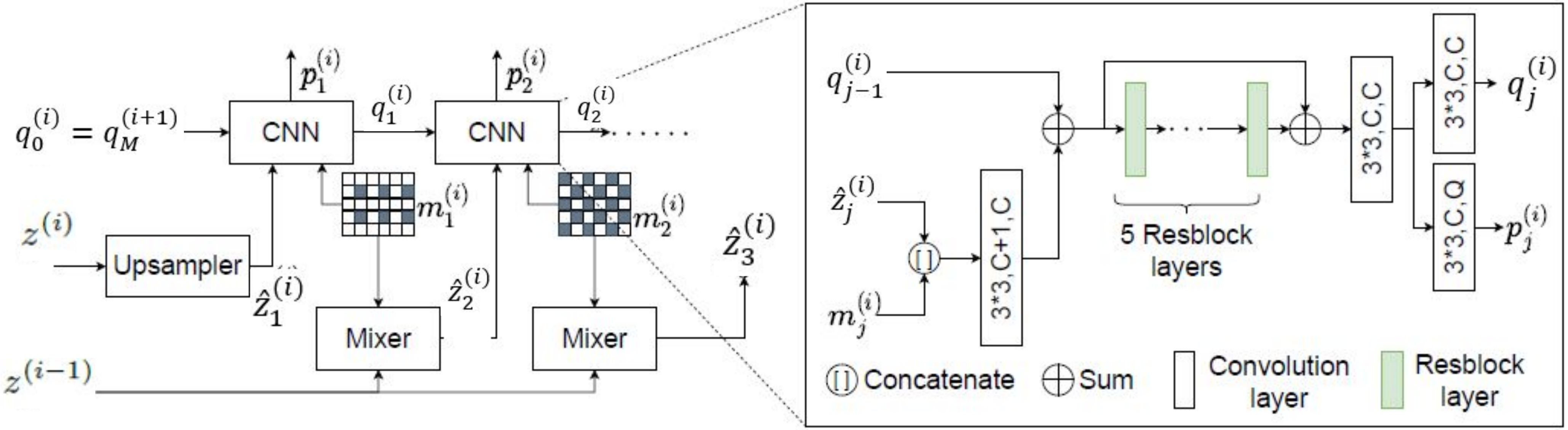
- *Partition each scale into multiple groups* ($g_j^{(i)}$: *group j at scale i*)
- *The context for group is* $C_j^{(i)} = \{g_1^{(i)}, g_1^{(i)}, \ldots, g_{j-1}^{(i)}, z^{(i-1)}\}$
- *The conditional distribution* $p(z^{(i-1)}|z^{(i)})$ *can be written as* $p\left(z^{(i-1)}|z^{(i)}\right) = \prod_{j=1}^{B_i} p\left(g_j^{(i)}|C_j^{(i)}\right).$
- z_k : *the element in group* $g_j^{(i)} \Longrightarrow$ $p\left(g_j^{(i)}|C_j^{(i)}\right) = \prod_{k=1}^{N_j^{(i)}} p\left(z_k|C_j^{(i)}\right).$

Proposed Method — L²C

B. Probability Model for Lossless Coding

1	2	1	2	1	2
3	x	3	x	3	x
1	2	1	2	1	2
3	x	3	x	3	x

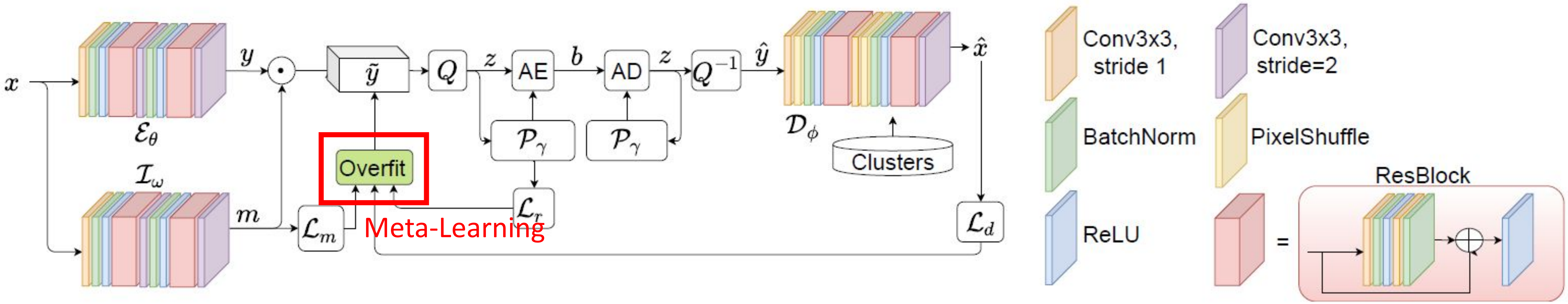




- $Assume\ p\left(z_k \middle| C_j^{(i)}\right)\ follows\ a\ mixture\ of\ logistic\ distribution$
- $The\ parameters\ are\ determined\ by\ a\ function\ modeled\ by\ a\ deep\ neural\ network\ with\ C_j^{(i)}\ as\ its\ input.$

Proposed Method — L²C

C. Meta-Learning



Proposed Method — L²C

C. Meta-Learning

- ***First Stage :***

✔ **Loss :**

$$\mathcal{L}(\hat{x}, x, z, \tau) = \lambda_{d_1}\mathcal{L}_{d_1}(\hat{x}, x) + \lambda_{d_2}\mathcal{L}_{d_2}(\hat{x}, x) + \lambda_{d_3}\mathcal{L}_{d_3}(\hat{x}, x) + \lambda_r\mathcal{L}_r(z) + \lambda_m\mathcal{M}(\tau)$$

- $\mathcal{L}_{d_1}: MS - SSIM\ Loss$
- $\mathcal{L}_{d_2}: Meam - squared\ error$
- $\mathcal{L}_{d_3}: Perceptual\ Loss$
- $\mathcal{L}_r: Probability\ module\ Loss$
- $M(\tau): Importance\ module\ Loss$

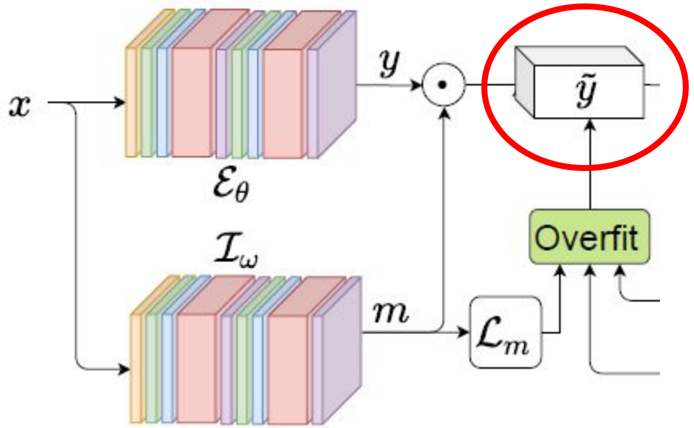
Proposed Method — L^2C

C. Meta-Learning

- ***Second Stage :***
  - ✔ the performance of latent tensor overfitting is maximized at inference time
  - ✔ Few-shots learning problem (1-shot)
    - allow for overfitting the latent tensor in few iterations

Proposed Method — L²C

C. Meta-Learning



- **Inner-Loop : Overfitting the latent tensor for each image x_i**
  - ✓ **Get the initial latent —** $\tilde{y}_i^{(0)} = \varepsilon_\theta(x_i) \odot m$
  - ✓ **The latent updated by gradient descent for n overfitting iterations and with learning rate α**

$$\tilde{y}_i^{(k+1)} = \tilde{y}_i^k - \alpha \nabla_{\tilde{y}_i} \mathcal{L}_{x_i}(\tilde{y}_i^k, \theta, \omega, \phi, \gamma)$$

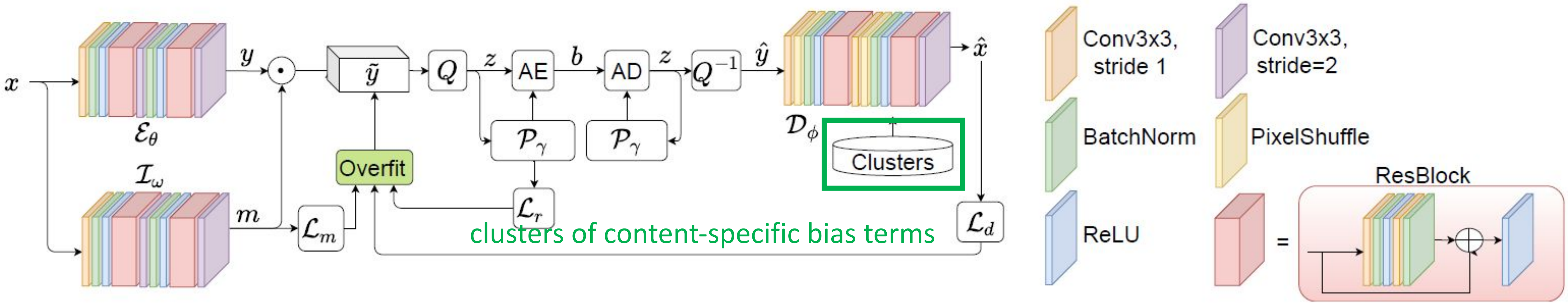
- **Outer-Loop : Overfitting the network parameters**
  - ✓ **Use overfitted latent tensor** $\tilde{y}_i^{(n)} = \varepsilon_\theta(x_i) \odot m$ **to compute the loss**
  - ✓ **Update the network's parameters with learning rate β**

$$\{\theta, \omega, \phi, \gamma\} = \{\theta, \omega, \phi, \gamma\} - \beta \nabla_{\{\theta,\omega,\phi,\gamma\}} \sum_{x_i \sim p(\mathcal{X})} \mathcal{L}_{x_i}(\tilde{y}_i^{(n)}, \theta, \omega, \phi, \gamma)$$

Proposed Method — L^2C

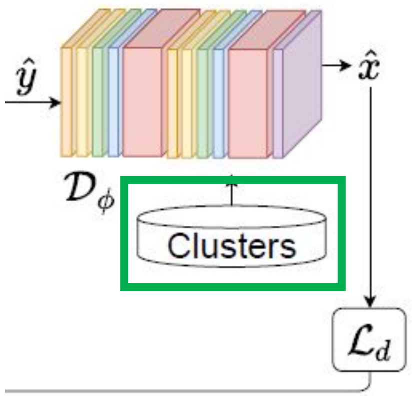
D. Adapting Decoders' Parameters

- ***Third Stage :***



Proposed Method — L^{2}C

clusters of content-specific bias terms



D. Adapting Decoders' Parameters

- Authors of [2] found that updating only the bias terms is a good trade-off between gain in reconstruction quality and bitrate overhead incurred by signaling the updated weights.
- overfit only the bias terms of the convolutional layers of $\mathcal{D}_\phi$
- 255 sets of overfitted decoder's parameters
- 1 index is reserved for default bias (no adaption)

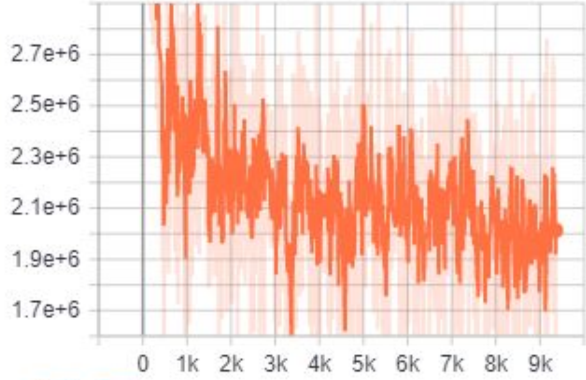
[2] Y.-H. Lam, A. Zare, F. Cricri, J. Lainema, and M. Hannuksela, "Efficient adaptation of neural network filter for video compression," arXiv: 2007.14267 [eess], 2020.

Experimental Results

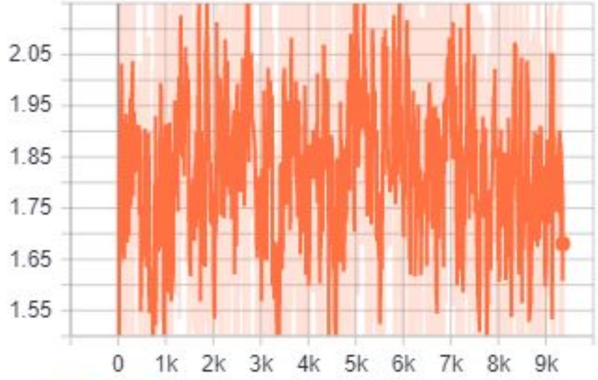
- # Quantization bits = 8
- msssim: 0.135 | mse: 21666.465 | bpp: 0.616 | PSNR: 25.134
- # Quantization bits = 6
- msssim: 0.161 | mse: 23985.810 | bpp: 1.460 | PSNR: 24.103
- # Quantization bits = 4
- msssim: 0.138 | mse: 22538.275 | bpp: 8.910 | PSNR: 24.714
- # Quantization bits = 2
- msssim: 0.341 | mse: 50103.955 | bpp: 926.334 | PSNR: 20.121

Quantization bits = 8

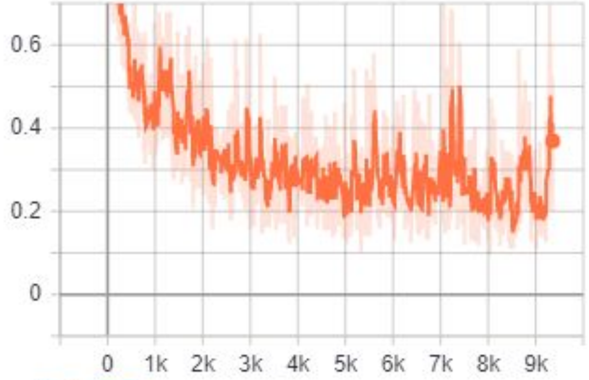
loss
tag: init_loss/loss



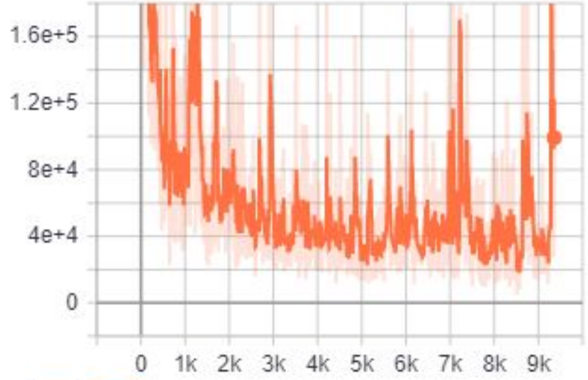
loss_M
tag: init_loss/loss_M



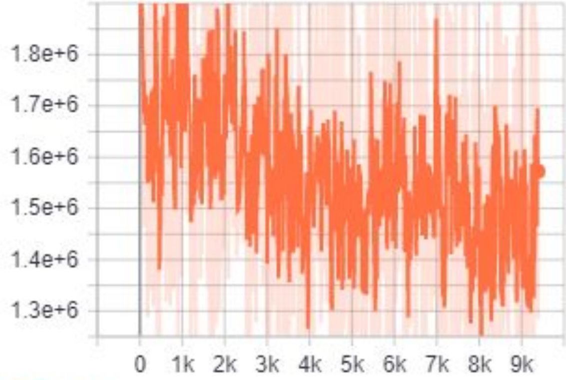
loss_d1
tag: init_loss/loss_d1



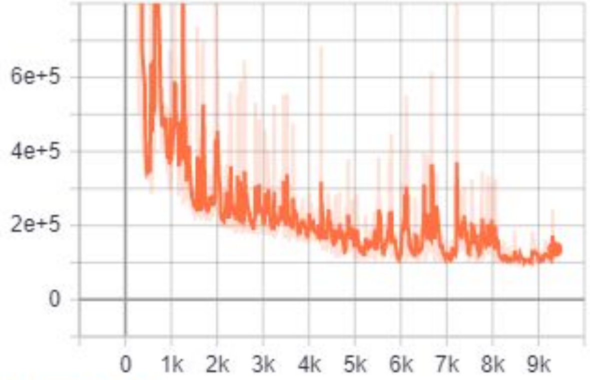
loss_d2
tag: init_loss/loss_d2



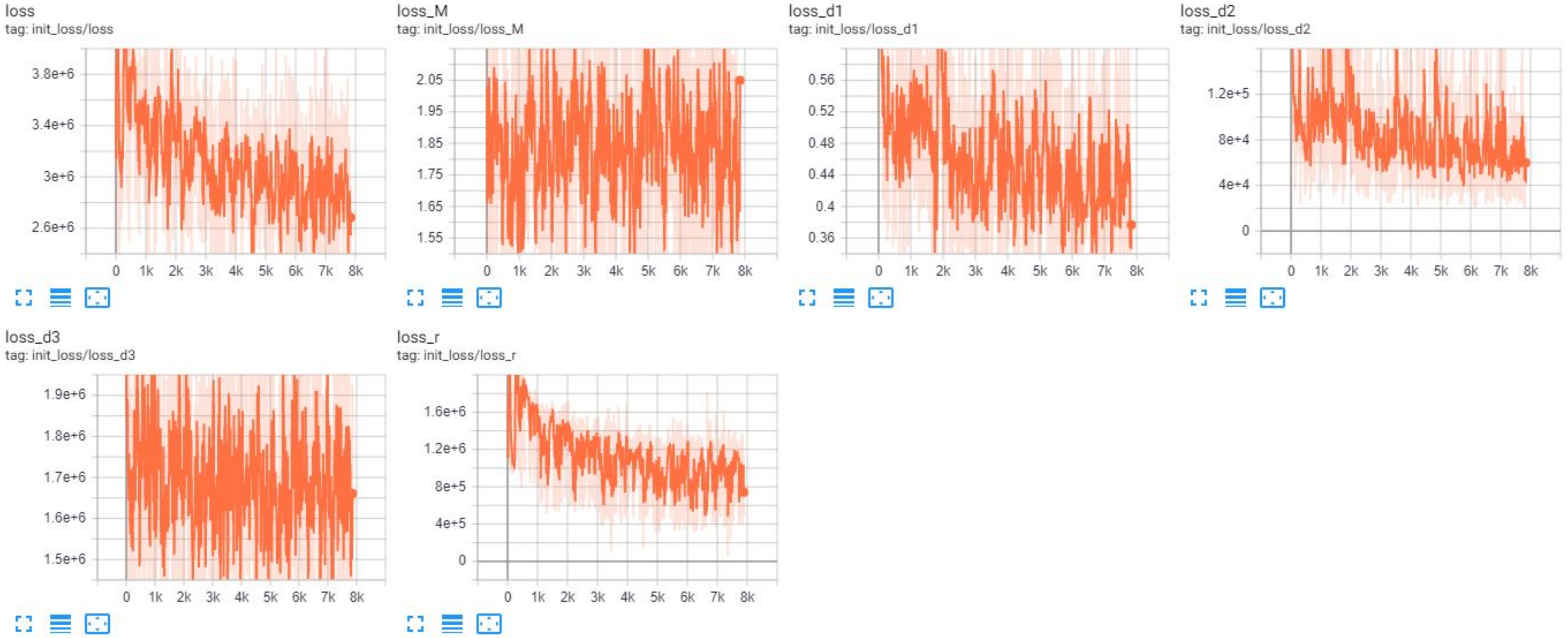
loss_d3
tag: init_loss/loss_d3



loss_r
tag: init_loss/loss_r

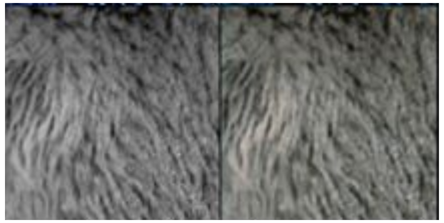


Quantization bits = 2



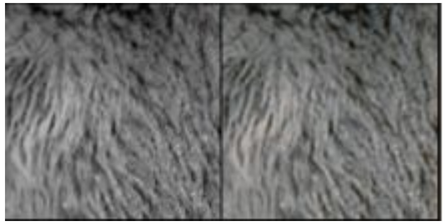
Experimental Results

- **# Quantization bits = 8**
- **MS-SSIM loss: 0.044 | MSE: 3668.470 | bpp: 0.552 | PSNR: 30.216**

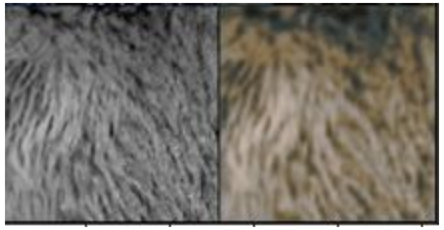


- **# Quantization bits = 6**
- **MS-SSIM loss: 0.103 | MSE: 6571.239 | bpp: 1.980 | PSNR: 27.685**

- **# Quantization bits = 4**
- **MS-SSIM loss: 0.035 | MSE: 3842.724 | bpp: 13.290 | PSNR: 30.007**



- **# Quantization bits = 2**
- **MS-SSIM loss: 0.505 | MSE: 19770.715 | bpp: 116.410 | PSNR: 22.878**



Experimental Results

- **# Quantization bits = 8**
- **MS-SSIM loss: 0.080 | MSE: 8635.454 |**

  bpp: 0.257 | PSNR: 26.481

- **# Quantization bits = 6**
- **MS-SSIM loss: 0.079 | MSE: 8980.742 |**

  bpp: 0.169 | PSNR: 26.320

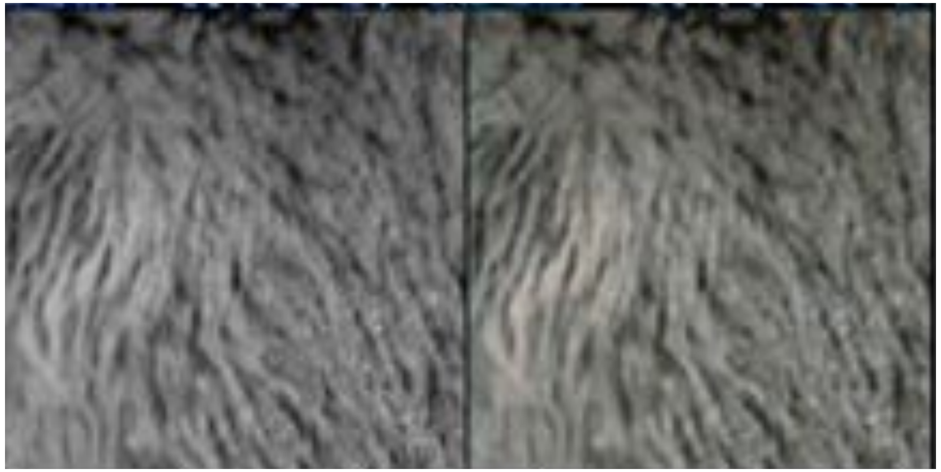
- **# Quantization bits = 4**
- **MS-SSIM loss: 0.084 | MSE: 8908.573**

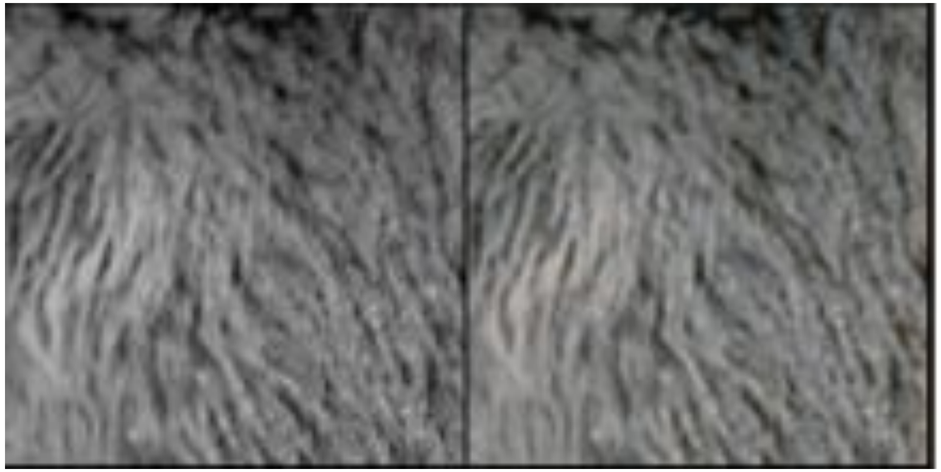
  bpp: 5.318 | PSNR: 26.348

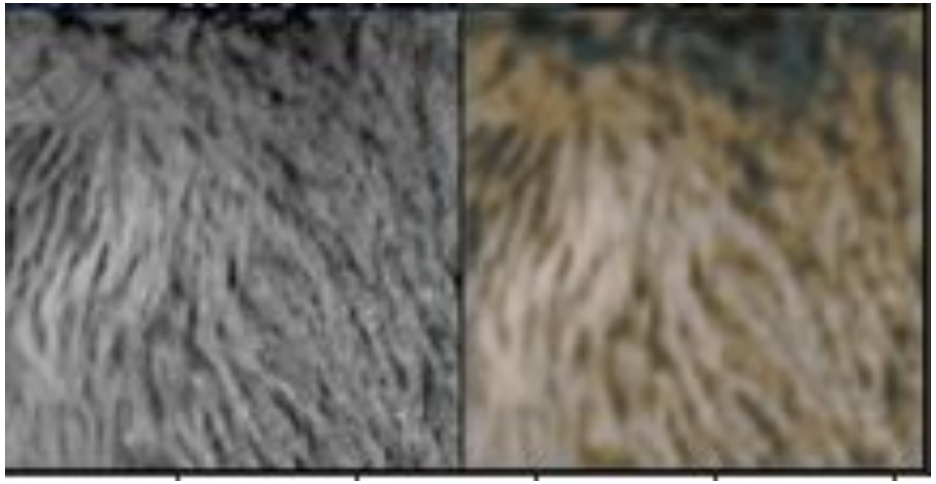
- **# Quantization bits = 2**
- **MS-SSIM loss: 0.248 | MSE: 27755.230**

  | bpp: 147.673 | PSNR: 21.414

Experimental Results







Experimental Results