|  |  |
| --- | --- |
| No. | **Neural Image Codec (NIC-0.1)** |
| Resource: | Editors |
| Title: | **Algorithm Discription of NIC Reference Software** |
| Date/Place: |  |



[http://www.ieee1857.org](http://www.ieee1857.org/)

<http://standards.ieee.org/develop/wg/AVS_1857_WG.html>



**Abstract**

NIC (Neural Image Codec) is the reference software of Future Video Coding Study Group (FVC-SG), IEEE Data Compression Standard Committee (DCSC). NIC aims at exploring the deep neural network based end-to-end solution for learned image compression. To this end, multiple image compression algorithms using deep learning will be extensively studied and investigated. This document is an official algorithm description about the latest version of NIC reference software.

Edit Notes:

NIC-0.1 and algorithm description:

* Incorporated VC-30-M256: the baseline model of NIC-0.1 is based on the proposal VC-30-M256 by Tong Chen, Haojie Liu, Ming Lu, Zhan Ma (Nanjing University).
* Incorporated the test protocol and NIC dataset.
* Incorporated splitting the image before encoding when the resolution is larger than 2K.

**1 Introduction**

At the 30th FVC-SG meeting (June 11, 2020, Online meeting), the SG decided to launch the investigation of deep end-to-end optimized image compression. Named as NIC, the based line model of reference software (NIC-0.1) was planned to release on June 25 2020.

**2 Method**

The proposed method mainly follow the design of Non-Local Attention optimization and Improved Context modeling-based image compression (NLAIC) [1] with modified Context Model as well as mixed Gaussian probability prediction [2].

## Overview of the encoding/decoding algorithm

The reference software uses 8bit RGB images as input, and output 8bit RGB reconstructions. No extra internal resolution change, bit-depth change and color space conversion are used.

## Overall Architecture

The network structure is based on [1] with overall structure as shown in Fig. 1. More details can be found in the paper, except that in this proposal we use modified Context Model with 3 mixed Gaussian probability predictions as described in [2].



Fig. 1 Overall structure of NIC-0.1 proposed network

The NIC framework is built on a variational autoencoder structure, with non-local attention modules (NLAM) as basic units in both main and hyperprior encoder-decoder pairs (i.e., Main Encoder, Main Decoder, Hyper Encoder and Hyper Decoder). Main Encoder with quantization Q are used to generate quantized latent features and Main Decoder decodes the features into the reconstructed image. Hyper Encoder and Hyper Decoder are applied to provide side information about the probability distribution of quantized latent features (known as hyperpriors), to enable efficient entropy coding. The hyperpriors as well as autoregressive spatial-channel neighbors of the latent features are then processed through the conditional context model P to perform conditional probability estimation for entropy coding of the quantized latent features.

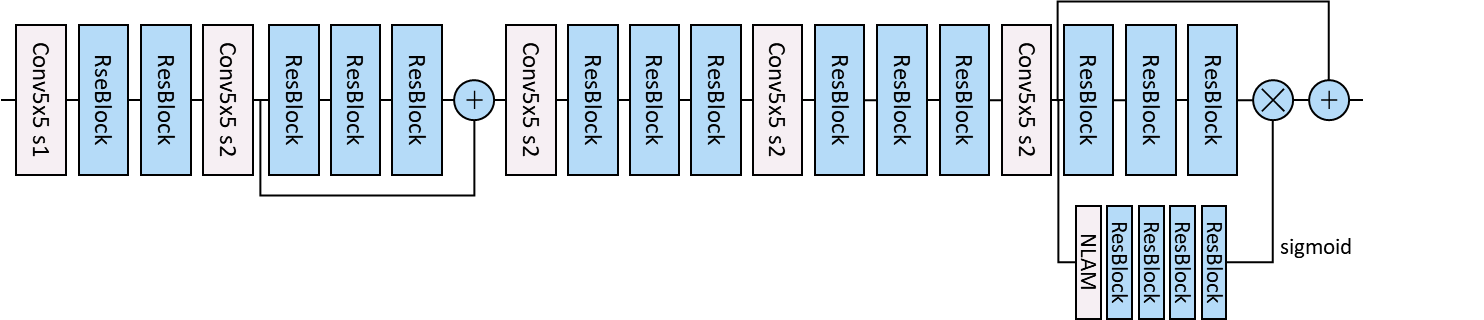


Fig. 2(a) Main Encoder Architecture

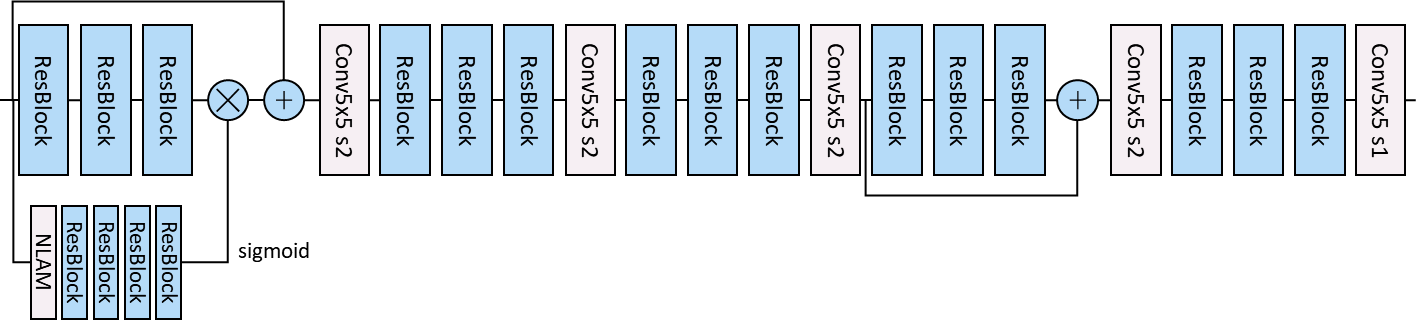


Fig. 2(b) Main Decoder Architecture

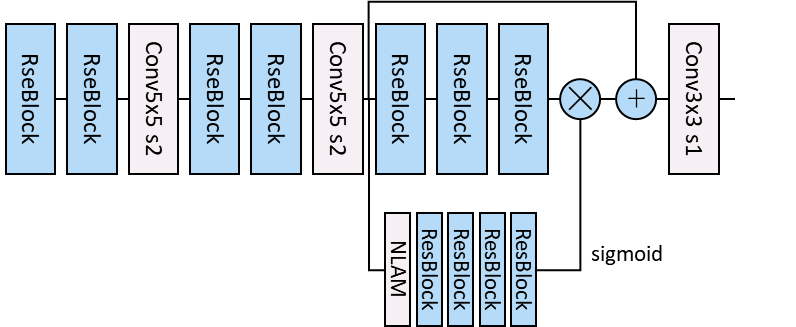


Fig. 2(c) Hyper Encoder Architecture

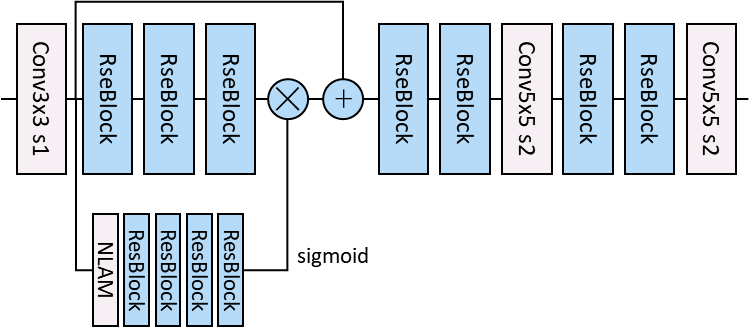


Fig. 2(d) Hyper Decoder Architecture

Fig. 2(a) to Fig. 2(d) show the detailed network architecture of the NIC Main Encoder/Decoder, Hyper Encoder/Decoder.

### Non-Local Module

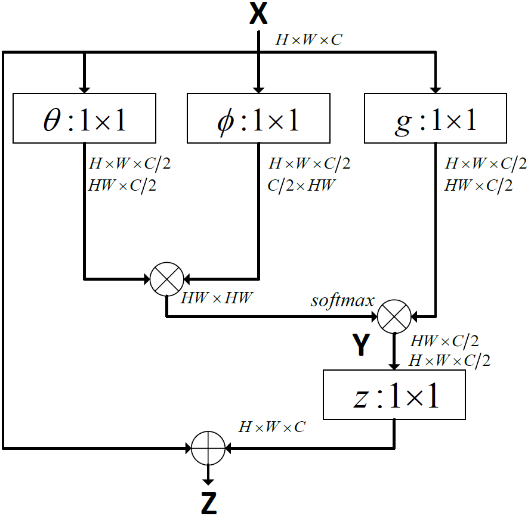


Fig. 3 Non-Local Module

Non-local operations are embedded into compression framework to capture both local and global correlations among the pixels in the original image and latent features. Attention mechanism is applied together with the non-local operations to generate implicit importance masks at various layers to guide the adaptive processing. These masks essentially help to allocate more bits to more important areas that are critical for rate-distortion efficiency.

### ResBlock

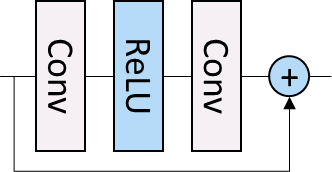


Fig. 4 ResBlock

ResBlock as shown in Fig.3 is used as the basic unit in the network with 2 Conv layers of kernel size 3.

### Probability Estimation with Context Model

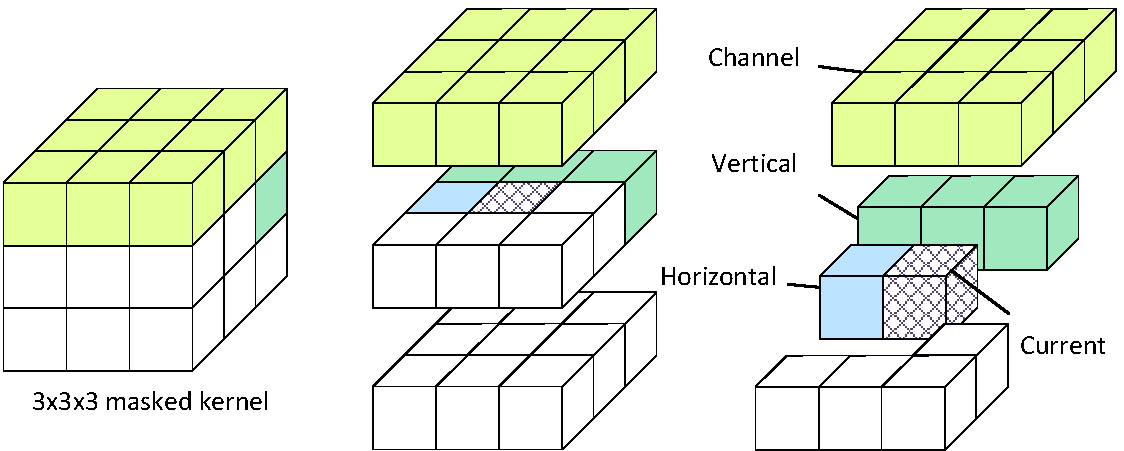


Fig. 5 3d-Mask CNN



Fig. 6 Context Model with GMM

To improve the context modeling of the entropy engine for better latent feature compression, a masked 3D CNN (i.e., 11×11×11) based prediction is used to approximate more accurate conditional statistics. Current pixel (in purple grid cube) is predicted by the causal/processed pixels (in yellow for neighbors from previous channel, green for vertical neighbors, and blue for horizontal neighbors) in a 3D space. Those unprocessed pixels (in white cube) and the current pixel are masked with zeros. The final probabilistic distribution can be measured by 3 weighted Gaussian distribution:

(1)

1. Head Syntax (cropping not included)

The syntax element of NIC is introduced as following table. To guarantee the correctness of decoding process. These syntax is essential including the image size and model index.

|  |  |  |
| --- | --- | --- |
| H  (uint16) | W  (uint16) | Model-Index  (uint8) |
| MAIN\_VALUE\_MIN  (int16) | MAIN\_VALUE\_MAX  (int16) | HYPER\_VALUE\_MIN  (int16) | HYPER\_VALUE\_NAX  (int16) |
| FileSize\_MAIN  (uint32) | | FileSize\_HYPER  (uint32) | |

## Methodology for training

### Quantization strategy

In this proposal, Universal quantization as described in [6] is used during training.

(2)

Here u is uniform noise within range [-1/2, 1/2].

### Loss function

MSE Loss function in RGB colorspace is used. The overall loss function is:

. (3)

Here D means distortion measured by MSE in RGB domain, and indicates estimated bitrate of main and hyper encoder.

### Training procedure

NIC train datasets are used for training and cropped to 256x256 patches. And random resize of original images before cropping is used for data augmentation. Adam [5] optimizer is used. Batch size is 12. Original learning rate is 5e-5, and will decay by 1/3 until 1e-6 when loss converged for previous learning rate.

### Bitrate allocation

λ is set to 400, 800, 1600, 3200, 6400 for MSE-RGB optimization.

### Model storage analysis

All models are trained and tested with FP32 precision and each model occupies 198.19MB + 2.9MB (Autoenocder + Context Model).

**3 Dataset**

A dataset is specified by NIC, which could be accessed at [https://www.bitahub.com/dataset/](https://www.bitahub.com/dataset/detail?active=1). The dataset contains four parts: training/training\_crop/validation/test. All of them are publicly available. It should be noted that the training\_crop/ folder is the resolution sliced version of train/ folder to boost the training speed or other requirement for training. The testset covers 5 different classes of resolutions to test the generalization ability and coding performance.

**4 Usage**

The training/validation process is based on the BitaHub platform. Since the current version of NIC still needs several third-party dependencies as well as float-point operations for neural network inference. Therefore, the docker image is used to make sure the results are reproducible and replicable. Using the following command could guarantee that the users are using the same test environment to prevent the decode inconsistency between different platform. (Please see Docker Installation for more details)

Docker image can be pulled from docker hub with pytorch1.3 and AE installed:

$ docker pull tongxyh/pytorch1.3\_ae:latest

Training:

$ python train.py

Testing:

Encode:

$ python inference.py –-encode –i <input\_image> -o <coded\_bin> -m <model\_index>

Decode:

$ python inference.py –-decode -i <coded\_bin> -o <rec\_image>

**5 Performance Evaluation**

This part is Rate-distortion (R-D) performance and encoding/decoding time on NIC test images, where distortion is measured by PSNR\_YUV and Y MS-SSIM, are summarized in Table I. The complexity is shown in Table II. The computing platform uses Bitahub. All encoding and decoding are performed on CPU.

## Objective results

The rate-distortion results generated by the NIC-0.1 codec are listed the following tables. results on the provided test set (not included)

Table I. Objective Results on Test Image ID\_XX (optimized for MSE RGB)

|  |  |  |
| --- | --- | --- |
| Metrics  bpp | PSNR\_YUV (dB) | Y MS-SSIM (dB) |
| 0.06 |  |  |
| 0.12 |  |  |
| 0.25 |  |  |
| 0.50 |  |  |
| 1.00 |  |  |
| 1.50 |  |  |

Encoding/Decoding Time is also shown in the following.

Table II. Complexity Test Image ID\_XX

|  |  |  |
| --- | --- | --- |
| Enc/Dec  bpp | Encoder  (s) | Decoder (s) |
| 0.06 |  |  |
| 0.12 |  |  |
| 0.25 |  |  |
| 0.50 |  |  |
| 1.00 |  |  |
| 1.50 |  |  |

# References

1. Neural Image Compression via Non-Local Attention Optimization and Improved Context Modeling, Tong Chen, Haojie Liu, Zhan Ma, Qiu Shen, Xun Cao, Yao Wang, arXiv preprint arXiv:1910.06244
2. An End-to-End Joint Learning Scheme of Image Compression and Quality Enhancement with Improved Entropy Minimization, Jooyoung Lee, Seunghyun Cho, Munchurl Kim, arXiv preprint arXiv:1912.12817
3. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study, Agustsson, Eirikur and Timofte, Radu, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2017
4. Microsoft COCO: Common Objects in Context, Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár, arXiv preprint arXiv:1405.0312
5. Adam: A method for stochastic optimization, Diederik P. Kingma, Jimmy Lei Ba, ICLR 2015
6. Variable Rate Deep Image Compression With a Conditional Autoencoder, Yoojin Choi, Mostafa El-Khamy, Jungwon Lee, arXiv preprint arXiv:1909.04802