

Supplementary Materials:

HybridFlow: Infusing Continuity into Masked Codebook for Extreme Low-Bitrate Image Compression

Anonymous Authors

1 TRAINING CONFIGURATIONS IN DETAIL

All of our training workload were launched over 8 NVIDIA V100 GPUs with 32GB GPU Memory each, using the AdamW optimizer ($weight_decay = 0.05$, $betas = (0.9, 0.95)$) and the CosineScheduler learning rate scheduler, and with the torch.cuda.amp.autocast (for faster training) turned on.

1.1 Mask Predictor Training

The hyperparameters were set as:

$batch_size : 10$
 $epochs : 20$
 $warmup_epochs : 2$
 $base_learning_rate : 5e - 4$
 $drop_out_ratio : 0.1$
 $gradient_clip = 3.0$

For the random mask ratio applied to the indices map d in this training stage, we followed the default setting of Masked Generative Encoder (MAGE), where the mask ratio was sampled from a truncated normal distribution of ($mean = 0.55$, $std = 0.25$) ranging within $[0.5, 1]$. Our mask predictor transformer can be viewed as a guided generation module, and it is important to enable enough generative capacity with high mask ratios (≥ 0.5). As indicated in the MAGE paper, masking out a huge portion of the indices map d would encourage the model to learn the internal semantic relationships between tokens within d , which helps to recover the masked indices map with just a small portion of the ground truth tokens.

We used the `LabelSmoothingCrossEntropy()` with smoothing = 0.1 as our loss function when calculating the prediction loss on the masked tokens, optimizing the generalization ability of the transformer predictor.

The original MAGE pre-training configuration had 400 total epochs and 40 warmup_epochs. Since we froze most of the weights in the pre-trained MAGE, and only trained the inserted CrossAttention module together with the affected MLP module within a single decoder block and the final output classification linear layer, we reduced the training workload to just around 1/20 of the original pre-training workload.

1.2 Pixel Decoder Training

The hyperparameter were set as:

$batch_size : 12$
 $epochs : 25$
 $warmup_epochs : 1$
 $base_learning_rate : 4e - 5$
 $gradient_clip = 3.0$

For the pixel distortion loss as described in Eq.2 in Section 3.3, the ratio of $w_1 : w_2$ for $L1_loss$ versus $LPIPS_loss$ was set as $8 : 1$, as the number of $L1_loss$ on pixels was generally 1/8 of the number of $LPIPS_loss$ using AlexNet. Such settings gave a proper balance between PSNR and LPIPS, which aligned with our target that the reconstructed images should pursue both faithfulness and perceptual quality.

2 BRIDGING THE CONTINUOUS FEATURES

As mentioned in Section 3.1, for the continuous-features based flow, we initially downsampled the input image by 4x and then fed it into the MLIC framework to get ultra-low bitrate. Thus, in actual implementation with default input image patchified as $256 \times 256 \times 3$, the latent \hat{y}_c had a shape of $8 \times 8 \times 320$. While the codebook-based latent \hat{y}_{VQ} had a shape of $16 \times 16 \times 256$ according to the model structure. Serving as the pixel correction network, our duplicate pixel decoder had the same network structure with the VQGAN pixel decoder, and to combine the continuous-domain and codebook-based information, \hat{y}_c was firstly transformed into the exact same size as \hat{y}_{VQ} before fed into the correction network. Also, \hat{y}_c was fed into the CrossAttention modules in Mask Predictor T for Key&Value computation, serving as the prediction assisting info. Therefore, \hat{y}_c was also transformed to match the dimension requirement for the transformer predictor.

There were several ways to conduct such transformations for dimension alignment, and we evaluated them as follows:

Test 1. Appended a linear layer of $[320, 768]$, transformed \hat{y}_c into prediction assisting info with size of $8 \times 8 \times 768$ and fed it into CrossAttention modules after flattening to $(64, 768)$. Appended an UpSampling Block with output channels of 256, transformed \hat{y}_c into pixel correction info with the size of $16 \times 16 \times 256$ and fed it into the correction network.

Test 2. Fed \hat{y}_c into the MLIC pixel decoding process, and generated a low-quality image restoration output of size $128 \times 128 \times 3$. The restored low-quality image was then resized via interpolation to the default size of $256 \times 256 \times 3$. Then a pretrained image encoder (4 Down-sampling Layers, with 256 output channels) took in the restored low-quality image and encoded it into a $16 \times 16 \times 256$ feature map. The feature map was directly fed into the correction network, serving as the pixel correction info and also fed into the CrossAttention module after channel linear transformation from 256 to 768, serving as the prediction assisting info.

Test 3. Fed \hat{y}_c into the MLIC pixel decoding process, and generated a low-quality image restoration output of size $128 \times 128 \times 3$. The restored low-quality image was then resized via interpolation to

the default size of $256 \times 256 \times 3$. After that, two separate image encoders were attached, one for extracting prediction assisting info and the other for extracting pixel correction info from the resized output image. Each encoder was trained with other modules for within its corresponding pipeline.

Evaluation of the above three methods showed that the last method provided a slightly better performance over the other two for both Mask Prediction and Image Correction (with the best training & validation loss).

3 MASK PREDICTION ACCURACY

The pre-trained MAGE had a converged training loss of about 5.6 on the ImageNet training set. After our continuous feature integration with the above method of **Test 3**, the training loss dropped to around 4.2 after 6 training epochs and converged to about 4.1 after entire training process.

We also evaluated the mask prediction accuracy directly by comparing the matching ratio between the predicted masked tokens and their corresponding ground-truth tokens.

With the mask schedule of "1_4" as shown in Fig.3 of the paper, where the token-based mask prediction aimed at predicting 192 (75%) masked tokens based on 64 (25%) ground-truth tokens, the original MAGE could only get an average matching ratio of about 5% on CLIC2020 datasets. In comparison, our mask predictor increased the matching ratio to over 20%. In Fig.4 of the paper, the first row showed the reconstructed images after pixel decoding with the single-stream pre-trained VQ-Decoder based on the mask schedule of "1_4" on indices maps. Results showed that the original MAGE preserved the shape distribution but had drastic color deviations (token matching ratio was only 3.7%), while our results were much better aligned with the ground-truth images in terms of both shape and color (token matching ratio reached 21.3%).

It is worth nothing that although the matching ratio of the masked tokens of the predicted indices map was only about 20%, the restored images still well aligned with the original ones. This is because the pixel decoder used \hat{y}_{VQ} for pixel generation, the overall semantic likelihood between the predicted indices map and the ground-truth mattered more than the absolute value of the token prediction.