Beyond pixels: CLIP-based Semantic Feature Compression

Ruiqi Shen

Information Processing and Communication Lab,
EEE Department, Imperial College London

2024.09

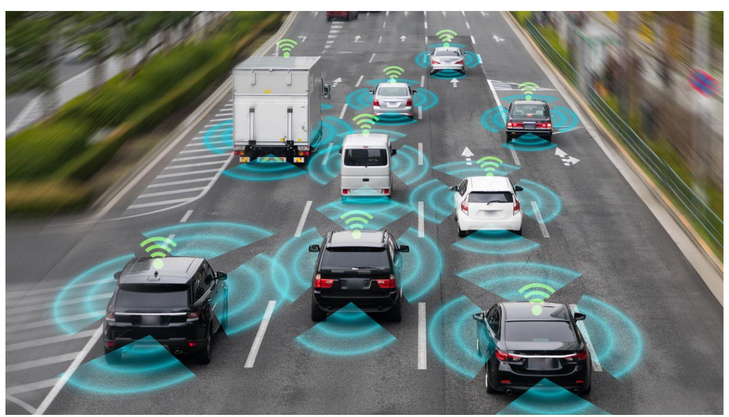
1. Introduction and Motivation

- Image compression: The quality of the reconstructed image (for local storage)

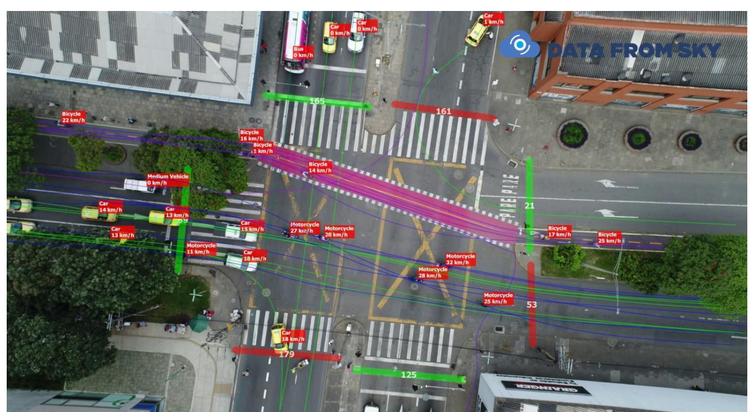
Problems:
- SOTA (0.1 bpp for acceptable image quality)
- Questionable generalization ability

- Semantic compression: The preservation of semantics (for source coding within digital transmission schemes)

Real-life scenes where semantic compression is needed:



Autonomous driving



Real-time Traffic monitoring

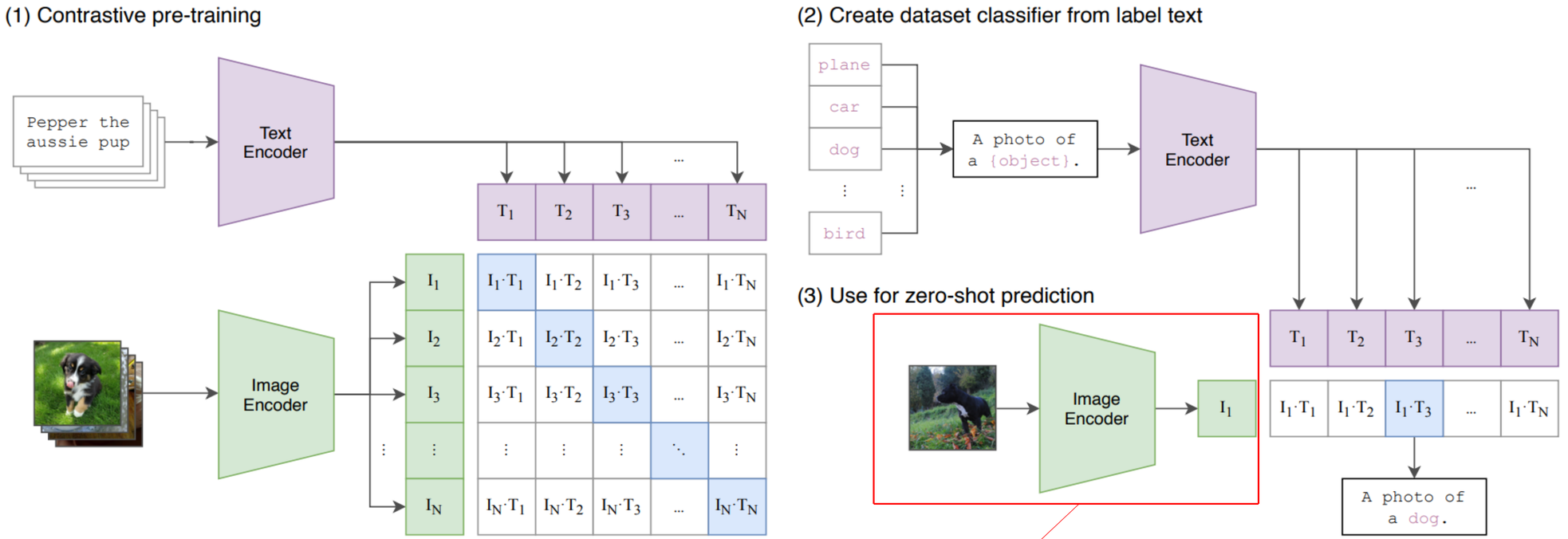
Drone Disaster Monitoring

(Drones send real-time fire images to data center, which analyzes the spread and extent to guide firefighting and evacuation)

Semantic compression:

- Receiver end: Semantics are needed instead of reconstruction
- Real-time → higher level of compression

1. Introduction and Motivation

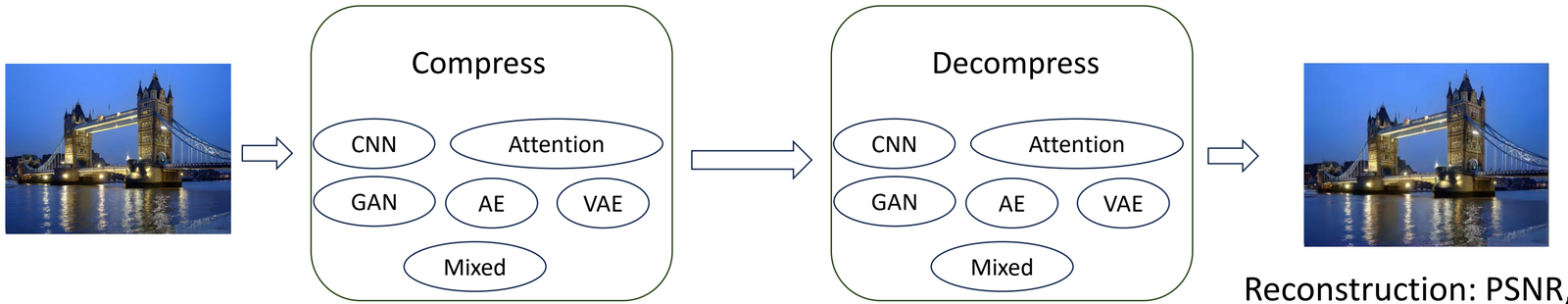


Left: How CLIP is trained ； Right: How CLIP could be used for zero-shot image classification

- CLIP is a large vision-language model which captures rich semantics
- When an image is input, the CLIP feature retains similar semantics but with significantly reduced dimensionality
- What if we further compress and quantize the CLIP features to see how many bits at least, are needed to represent such feature?

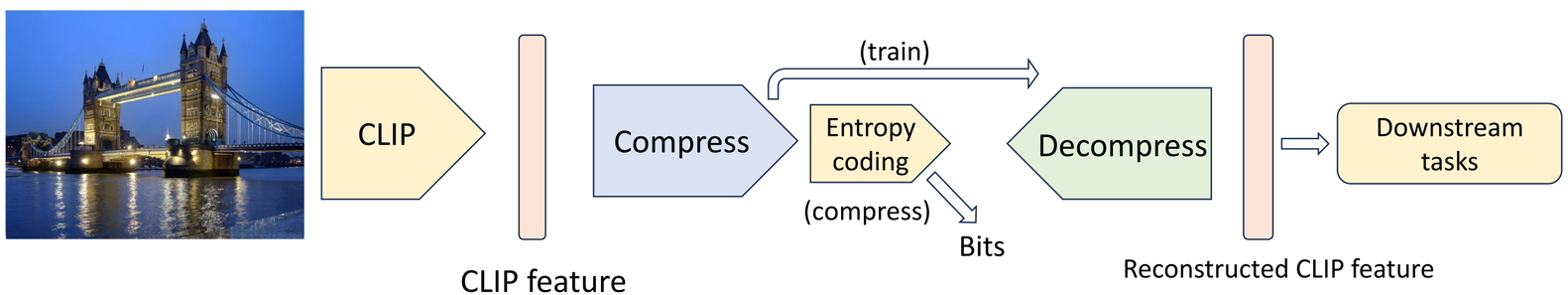
1. Introduction and Motivation

Image compression methods:



Reconstruction: PSNR, MS-SSIM
Compression: bpp(bits per pixel)

Semantic CLIP-feature compression (Our method):



Reconstruction of CLIP feature: cosine similarity
Compression: bpd (bits per CLIP feature dimension)

Task-agnostic Training: Focus on the current CLIP features only

1. No other samples
2. No other modalities of the same sample

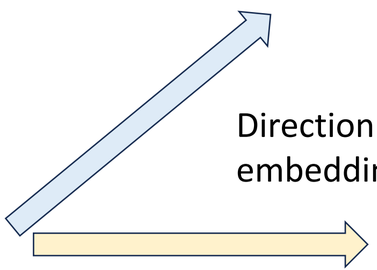
2. Methods

2.1 Cosine Similarity

During training, use cosine similarity to evaluate how well the semantics have been preserved within the compressed CLIP feature.

MSE: ≈0
Cosine Similarity: 0.4160

MSE: 0.3281
Cosine Similarity: 0.8320



Direction within CLIP's high-dimensional embedding space is very important

For two CLIP feature vectors a and b:

$$\text{Cosine Similarity}(a,b) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$$

Cosine similarity represents the direction similarity within the high-dimension space, which is exactly what we want for evaluating semantic preservation.

2. Methods

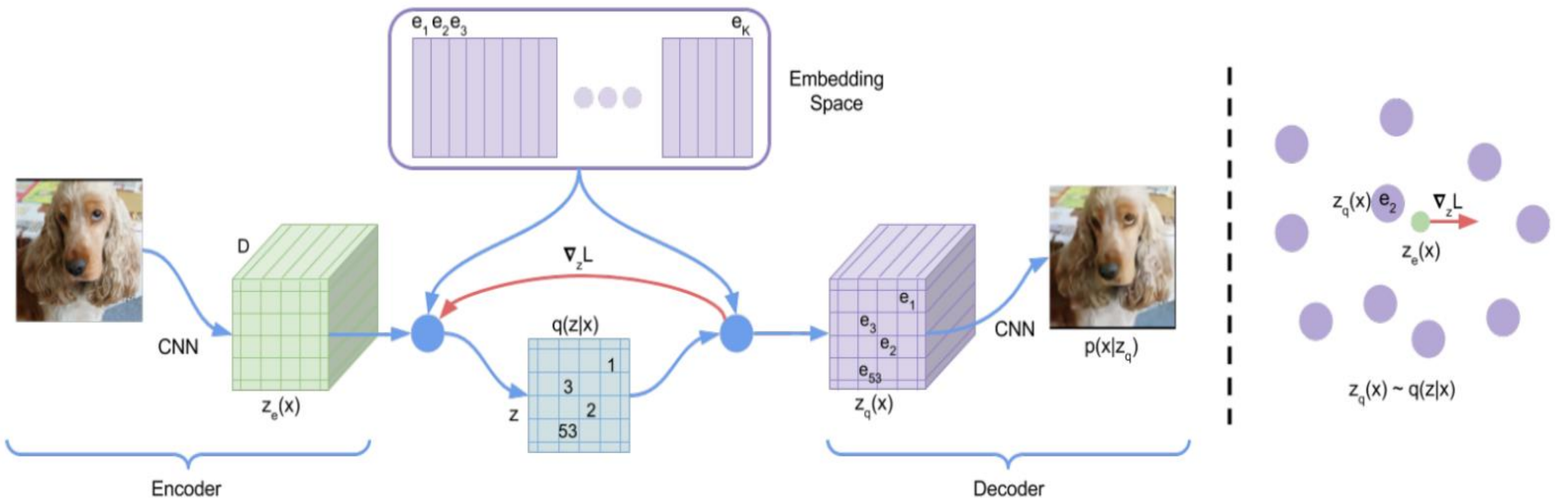
2.2 PQ-VAE with shared codebook

(Product-quantization VQ-VAE with shared codebook)

Quantization is a essential within the compression process

Why VQ-VAE-based method for CLIP feature compression:

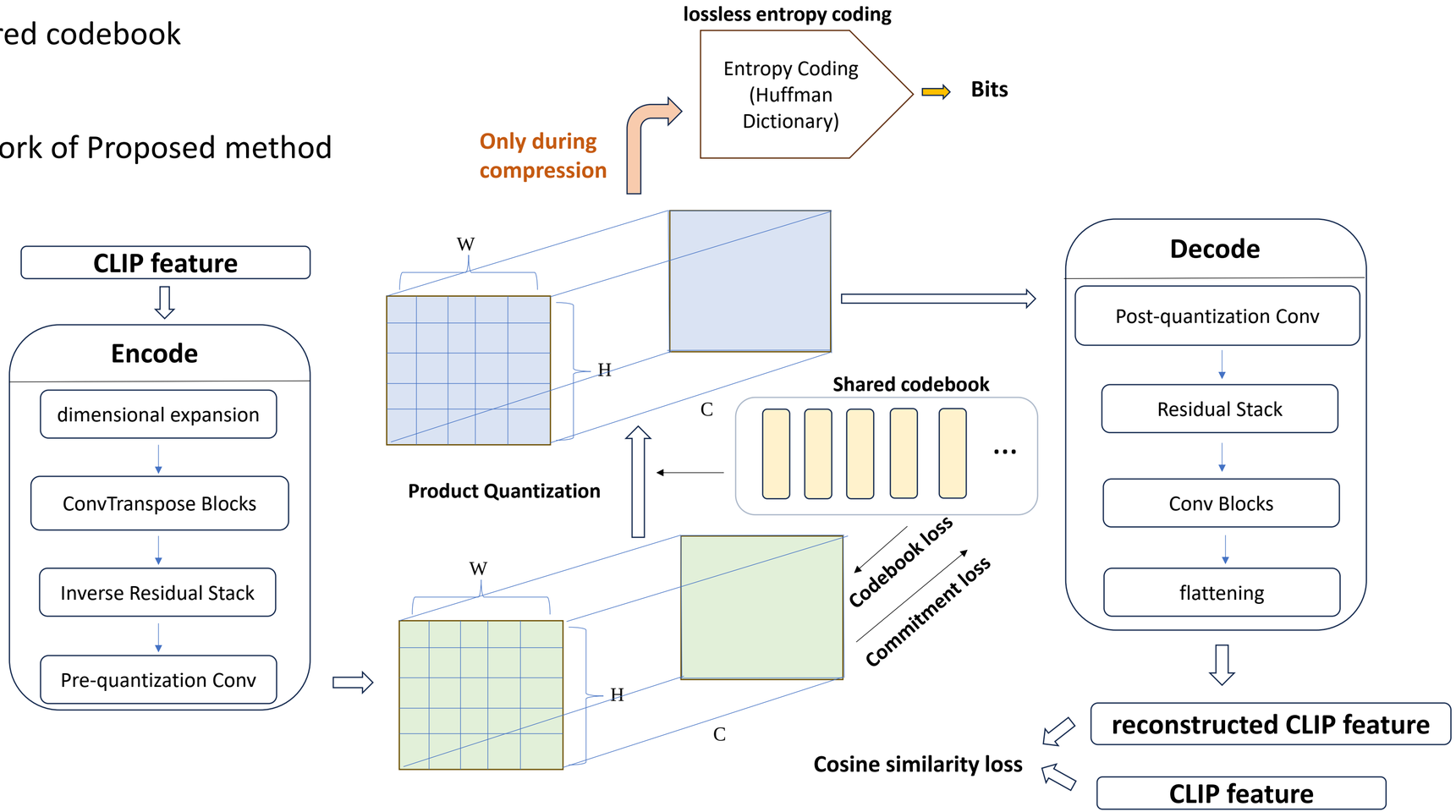
1. Inherent Vector-Quantization Capability
2. Generative Alignment with CLIP
3. Learnable Quantization Framework

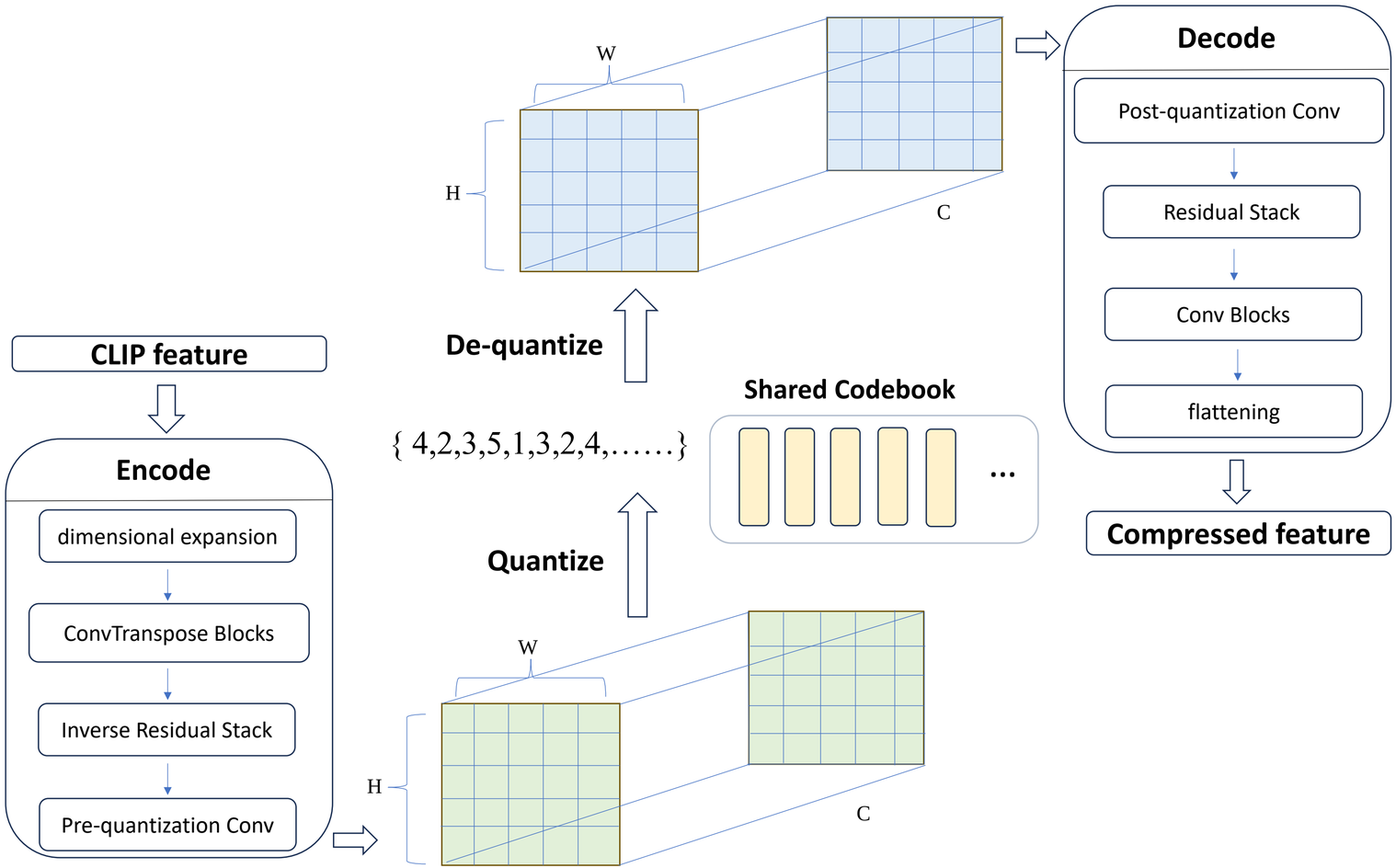


2. Methods

2.2 PQ-VAE with shared codebook

- Overall framework of Proposed method




CLIP feature
Encode
dimensional expansion
ConvTranspose Blocks
Inverse Residual Stack
Pre-quantization Conv
Quantize
{ 4,2,3,5,1,3,2,4,……}
Shared Codebook
...
De-quantize
W
H
C
Decode
Post-quantization Conv
Residual Stack
Conv Blocks
flattening
Compressed feature

2. Methods

2.2 PQ-VAE with shared codebook

- The quantized input has H*W feature channels
- **Each high-dimension feature channel is decomposed into n low-dimension subspaces, each subspace corresponds to a sub-vector**
- The codebook is shared across all channels and subspaces
- The codeword is designed to have the same dimension of each sub-vector

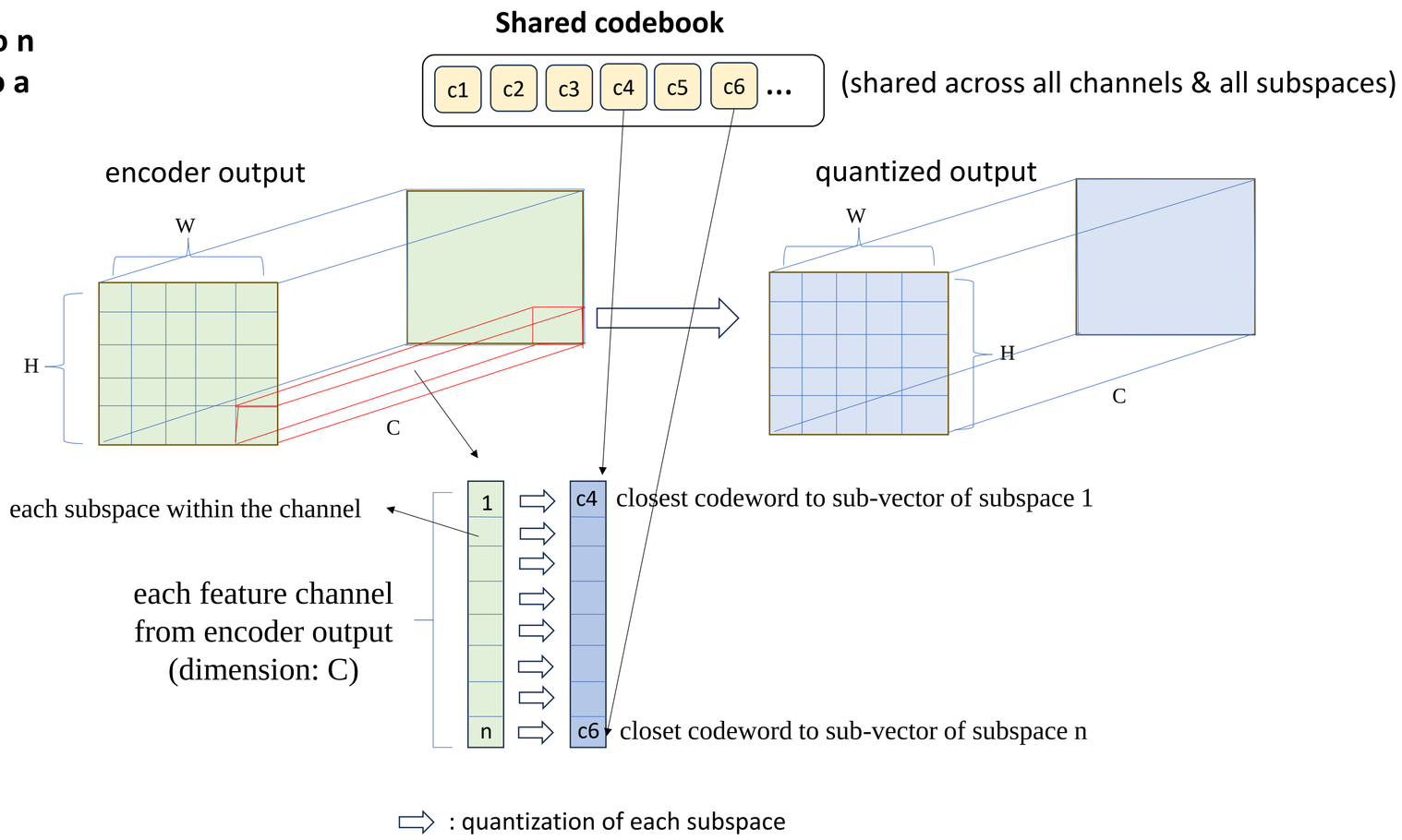
Algorithm Product Quantization using Shared Codebook (The quantization part)

Input: Encoder output tensor $\mathcal{T}$ with dimensions $H \times W \times C$, Shared Codebook $\mathcal{C}$

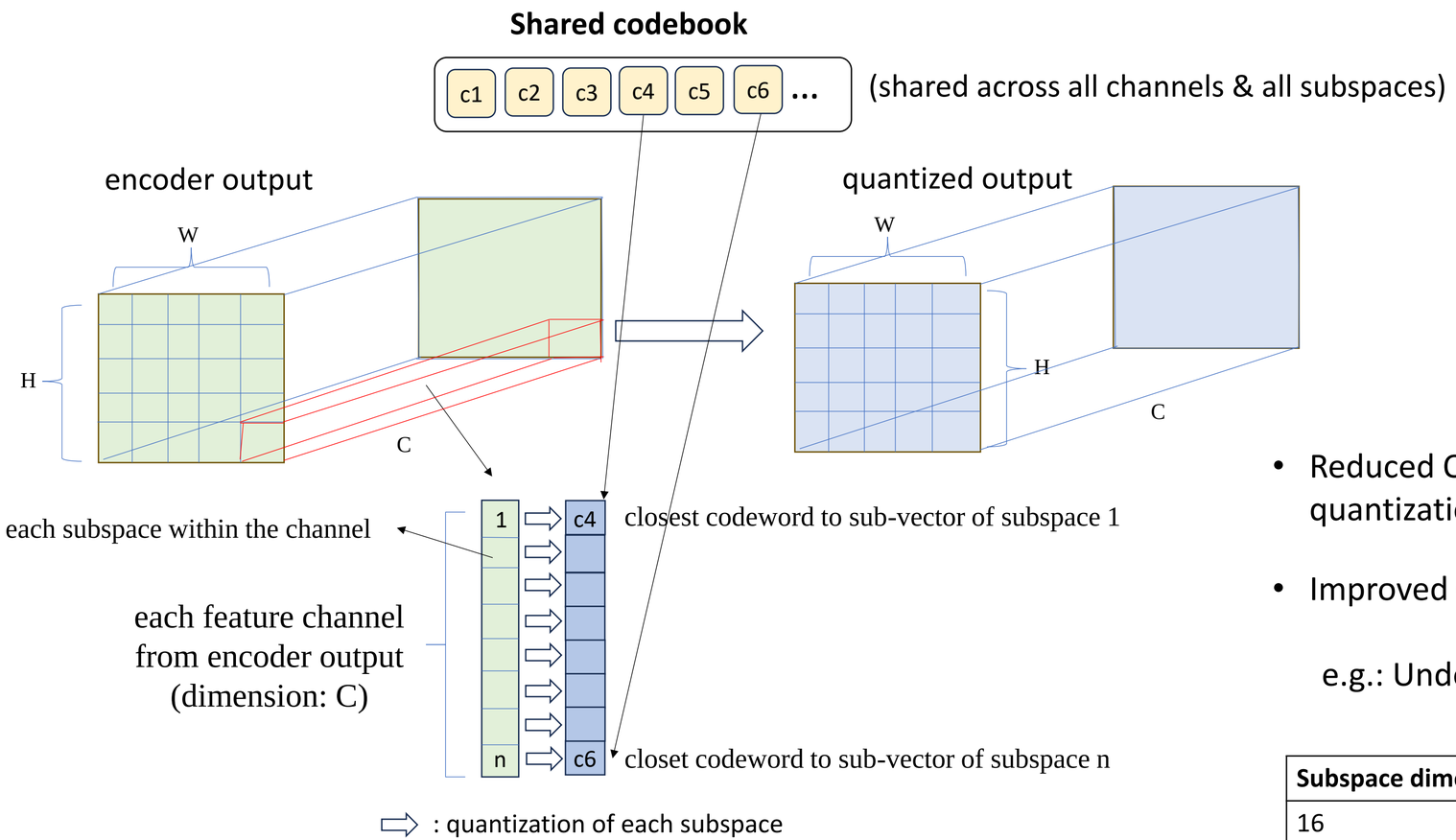
Output: Quantized output tensor $\mathcal{Q}$ with dimensions $H \times W \times C$ representing the original encoder output in terms of the shared codebook

for each feature channel $c \in [1, H * W]$ **do**
  $\mathbf{f}_c \leftarrow$ Extract feature channel c from tensor $\mathcal{T}$
  Step 1: Subspace Partitioning
  Divide $\mathbf{f}_c$ into n subspaces: $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$
  for each subspace $\mathbf{s}_i$ in $\mathbf{f}_c$ **do**
    Step 2: Sub-vector Quantization for Each Subspace
    Compare sub-vector $\mathbf{s}_i$ with all codewords in the shared codebook $\mathcal{C}$
    Find the closest codeword $c_j \in \mathcal{C}$ for sub-vector $\mathbf{s}_i$ using the nearest neighbor principle
    Quantize subspace $\mathbf{s}_i$ to codeword c_j
  end for
  Step 3: Formation of Quantized Output
  Combine the quantized results from all subspaces $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$ to form the quantized output $\mathbf{q}_c$ for channel c
end for
Output: Quantized output tensor $\mathcal{Q}$ with dimensions $H \times W \times C$

Details of product quantization with shared codebook



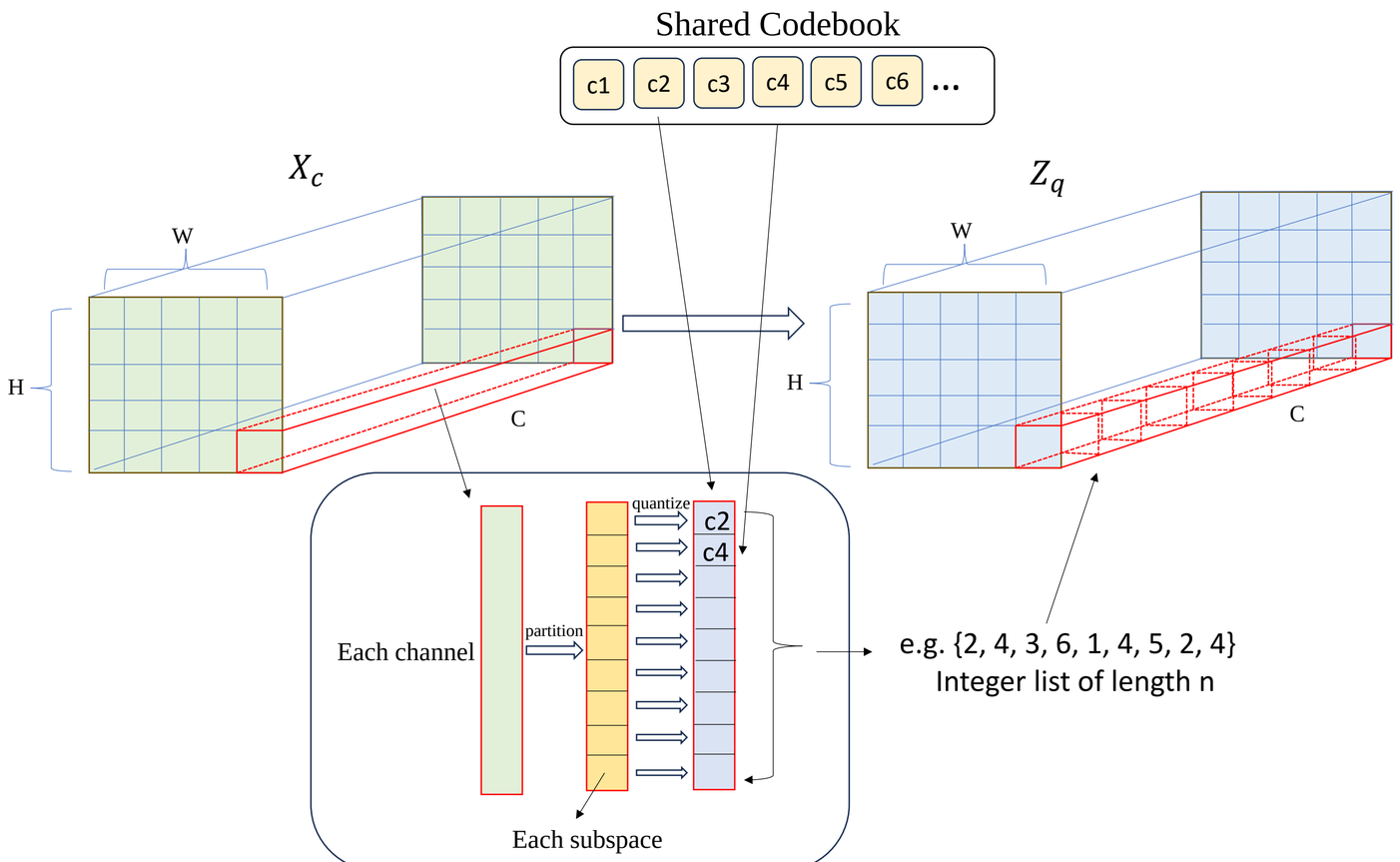
- Detailed animation display of the quantization process



- Reduced Quantization Error: Smaller subspaces enable finer-grained quantization
- Improved Codebook Efficiency and Prevention of Codebook Collapse:

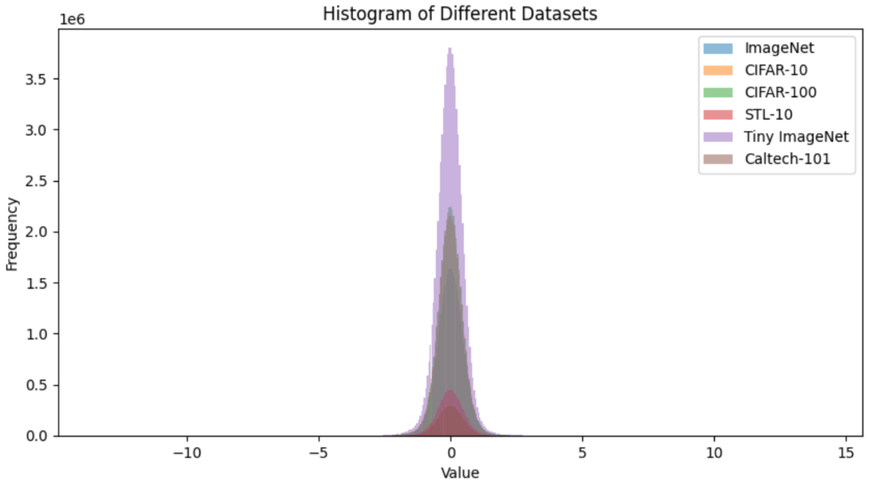
  e.g.: Under 600 bits constraint, H=W=5, C=128:

Subspace dimension	**Codebook size**
16	8
128 (original)	16777216 (2^{24})


Shared Codebook
c1
c2
c3
c4
c5
c6
...
X_c
W
H
C
Z_q
W
H
C
quantize
c2
c4
Each channel
partition
Each subspace
e.g. {2, 4, 3, 6, 1, 4, 5, 2, 4}
Integer list of length n

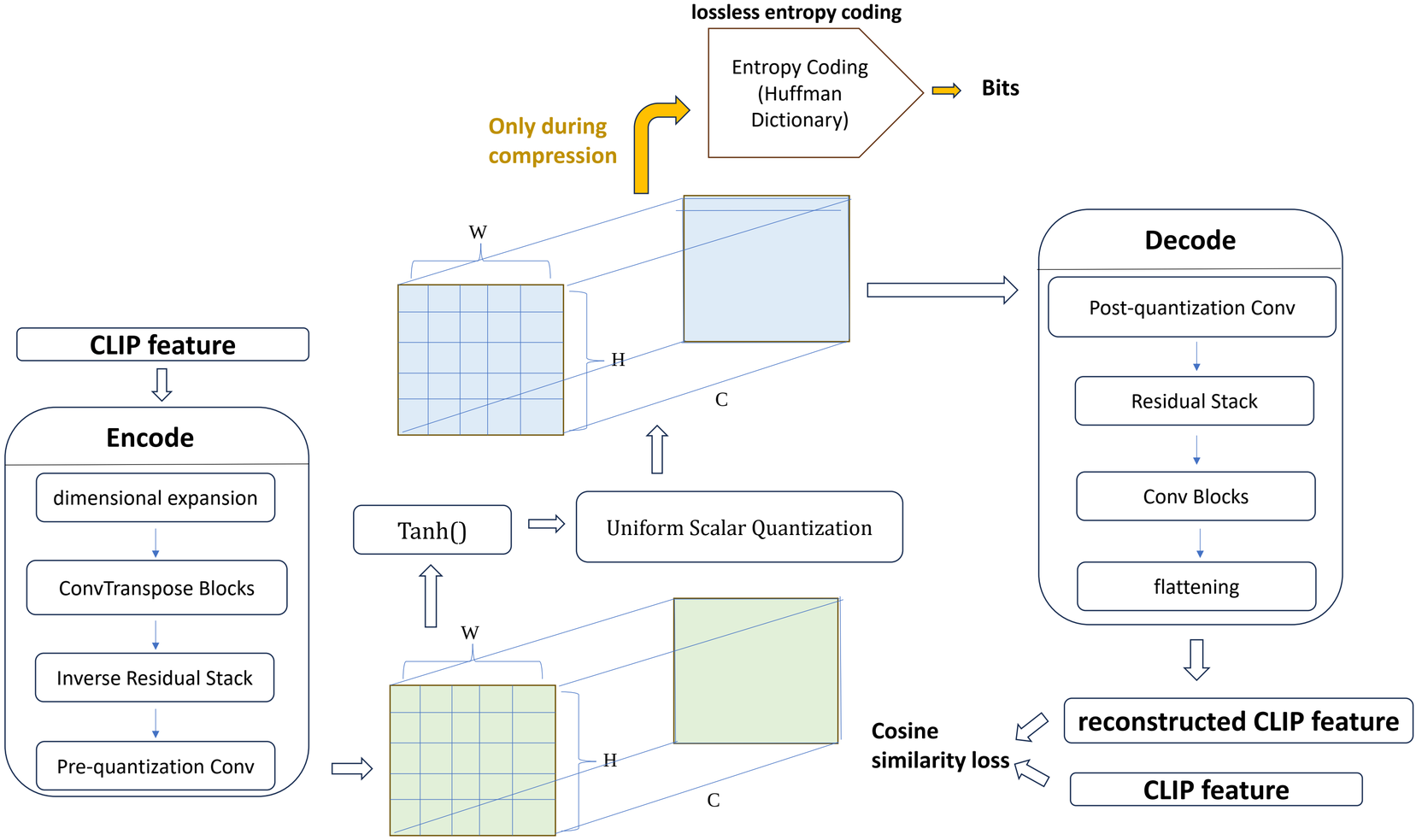
2. Methods

2.3 Autoencoder-based scalar quantization



Distribution of CLIP Feature Entries Across All Datasets

Range	Percentage (%)
(-1,1)	95.15
(-0.8, 0.8)	90.27
(-0.6, 0.6)	80.17
(-0.4, 0.4)	61.84
(-0.2, 0.2)	34.18



3. Evaluation

3.1 Validation Method and Metrics

- Semantic accuracy: Test accuracy of zero-shot image classification (classification accuracy between compressed CLIP image feature and all CLIP text features)
  Text prompt format: "a photo of a {class}"

  Calculation:
  $similarity\_score\_matrix = image\_features * text\_features^t$
  $predicted_i = argmax_j(similarity\_score\_matrix_{i,j})$
  $test\_accuracy = (predicted = labels)/sample\_numbers$

- Compression ratio: bpd (bits per CLIP feature dimension)
  $bpd = \frac{bits\ number}{768}$, (768 the dimension for each CLIP feature)

Configuration	Details
Dataset[1]	ImageNet
Train Classes	800 classes
Validation Classes	200 classes (different from training classes)
Compression Model	PQ-VAE with shared codebook, Autoencoder-based scalar quantization
Compression Metric	Bits per CLIP feature dimension(bpd)
Semantic Metric[2]	Zero-shot test accuracy of image classification
CLIP Model	ViT-L/14-336px (Frozen)
Reconstruction Loss	Cosine Similarity Loss
Quantize Loss	Codebook Loss, Commitment Loss (only for PQ-VAE with shared codebook)
Entropy Coding	Huffman coding (if not otherwise specified)

1. We will demonstrate our compression model's zero-shot performance of image classification on datasets including CIFAR10, CIFAR100, STL10, Tiny-ImageNet

2. Classification is only used for validation process for level of semantic preservation, no relation to the task-agnostic training

---

For both methods, multiple configurations are available under a certain bpd constraint:
e.g. for a bpd constraint of 0.78 (≡600 bits), all of the following configurations on the right are possible:

Codeword Dimension	Codebook size
16	8
32	64
64	4096

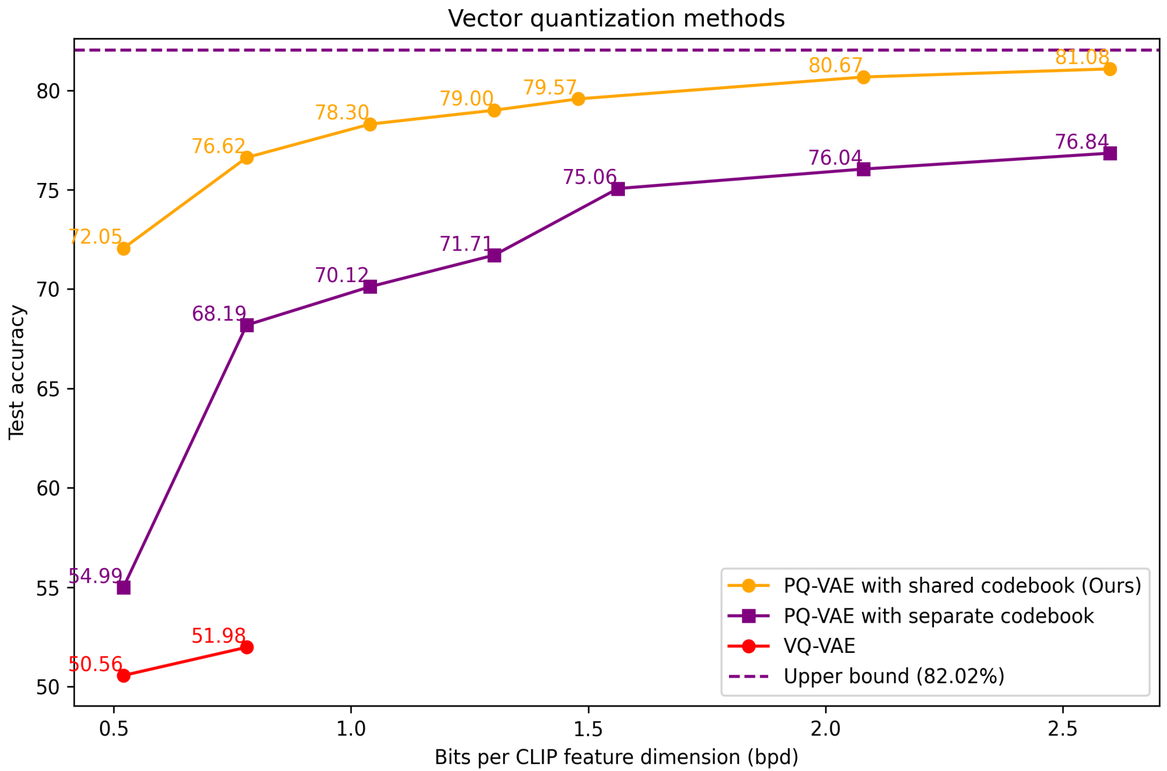
PQVAE with shared codebook at 600bits

Encoder Output Dimension	Quantization Levels
8	8
12	4
24	2

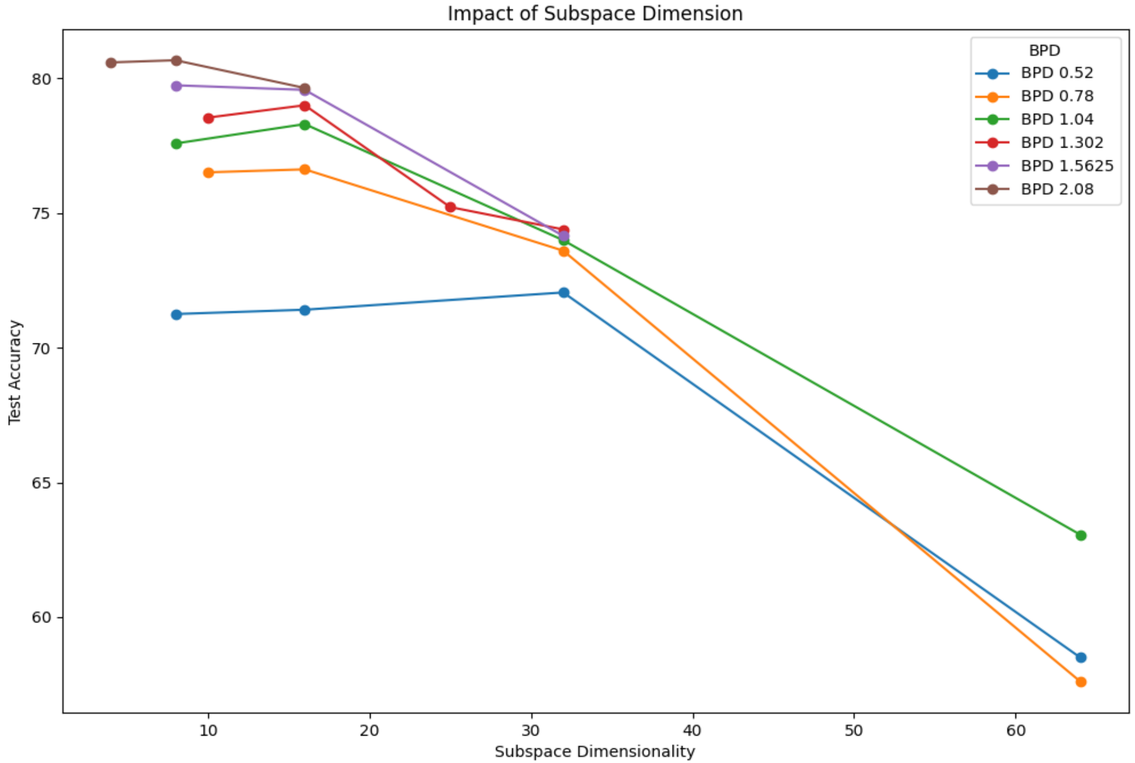
Autoencoder-based scalar quantization at 600bits

3. Evaluation

3.2 Results for vector quantization methods



PQ-VAE with shared codebook is compared with two vector quantization baselines



Ablation study: Different subspace dimensions under same bpd constraint

- Finer-grained quantization with reduced error
- Improved Codebook Utilization
- Prevention of Codebook Collapse

e.g.: At the same bpd constraint of 0.78bpd (≡600 bits),

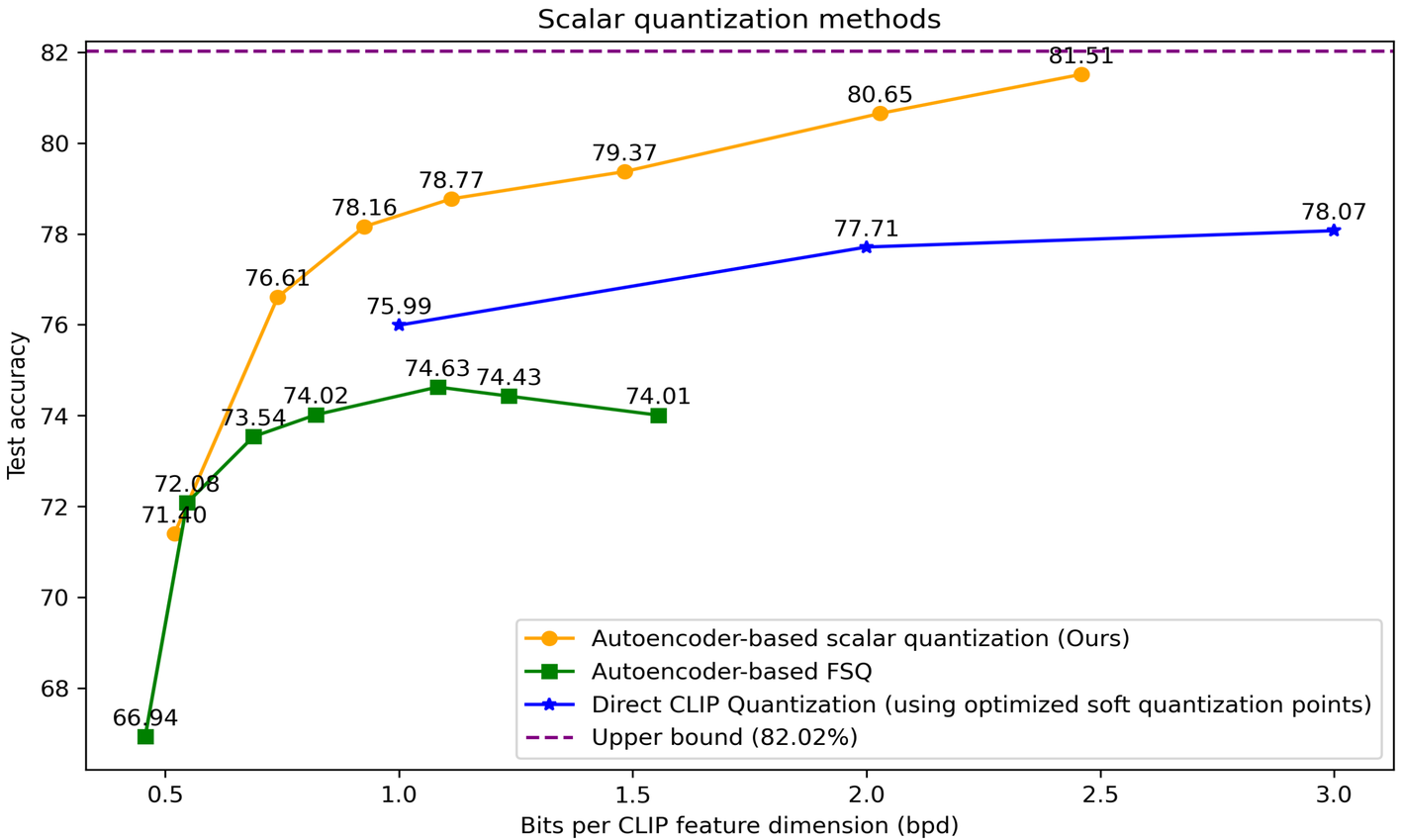
Subspace dimension	**Codebook size**
16	8
128 (original)	16777216 (2^{24})

3. Evaluation

3.3 Results for scalar quantization methods

bpd	1 (768 bits)	2 (1536 bits)	3 (2304 bits)	4 (3072 bits)	5 (3840 bits)
Test accuracy	75.99%	77.71%	78.07%	81.73%	81.95%

Results of **direct** scalar quantization on CLIP features **(blue line)**



Autoencoder-based scalar quantization is compared with two baselines

Direct scalar quantization:

- Quantization points are optimized using softmax-based soft quantization with temperature = 1
- Initialization: uniform distributed between (-0.5, 0.5)

FSQ (Finite Scalar Quantization)
(SOTA method which claims to outperform VQ-VAE)

- $q(z_i) = \text{round}(\lfloor \frac{L}{2} \rfloor * \tanh(z_i))$ (z_i: quantize input)

Problems with direct scalar quantization:

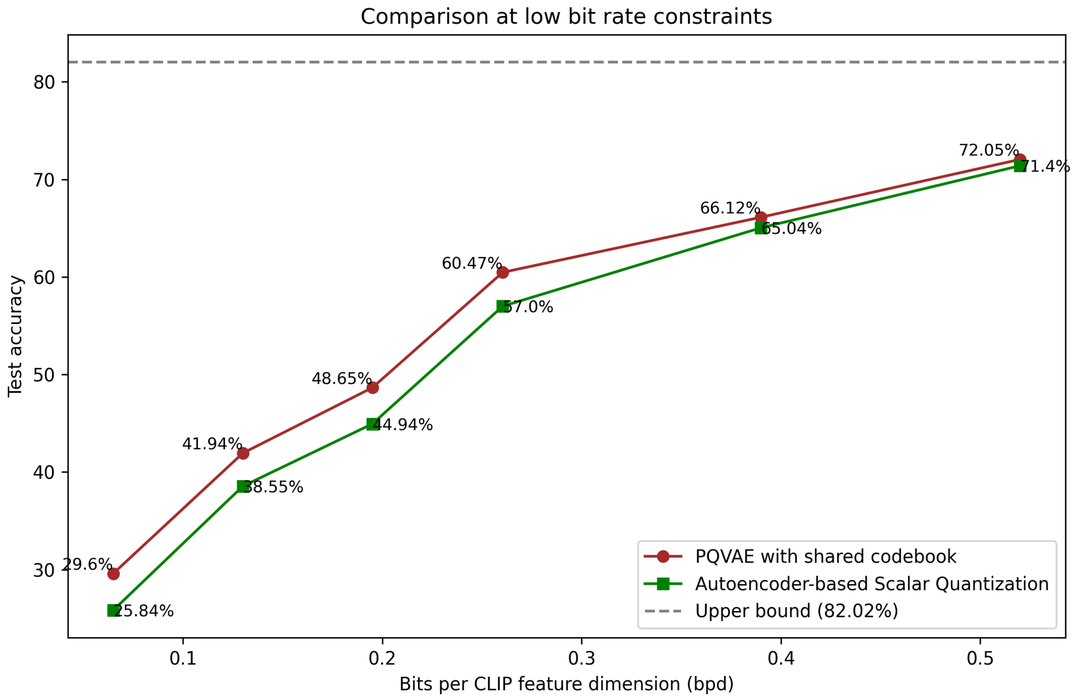
- Slow improvement with pretty high cost of bits
  (An additional 768 bits are needed for each increment)

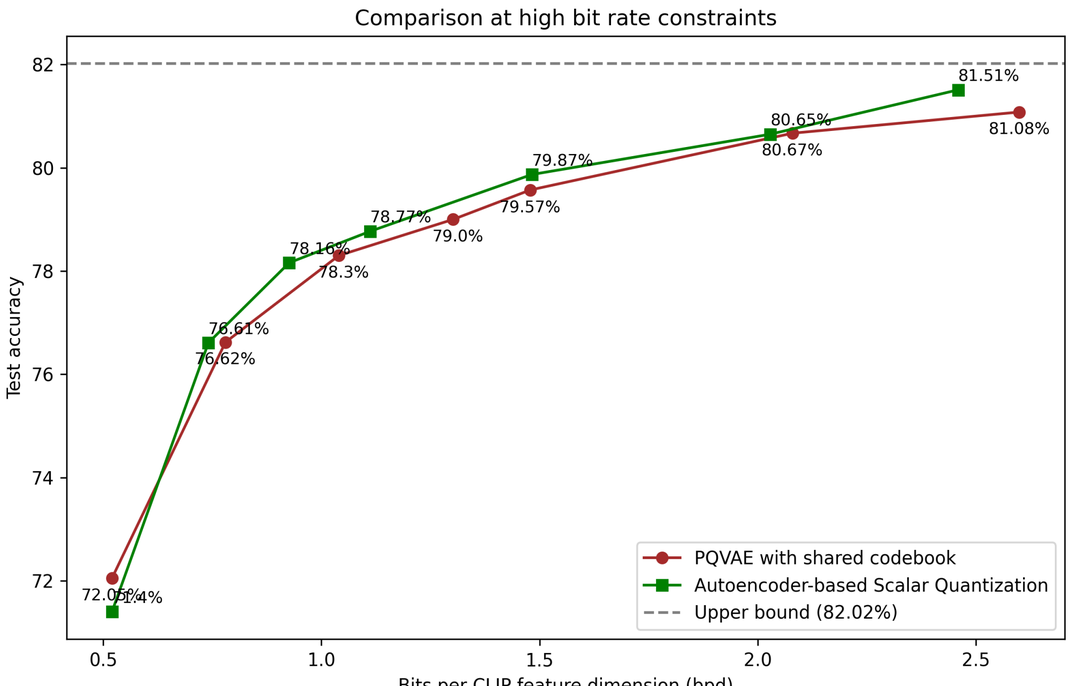
Advantages of Autoencoder-based scalar quantization:

- Dimension reduction of feature channel dimension
- Higher configuration flexibility under the same bit constraint
  (balance between encoder output dimension & quantization levels)

3. Evaluation

3.4 Comparison between the two proposed methods





- Low bit rate constraints (50→600bits): PQ-VAE with shared codebook performs better by **adjusting only the codebook parameters** (codebook size, codeword dimension), the shape of feature maps representing CLIP feature is fixed.

- High bit rate constraints (600→2000bits): Autoencoder-based scalar quantization performs better with **direct adjustment on the feature map** (encoder output channel dimension, quantization levels)

3. Evaluation

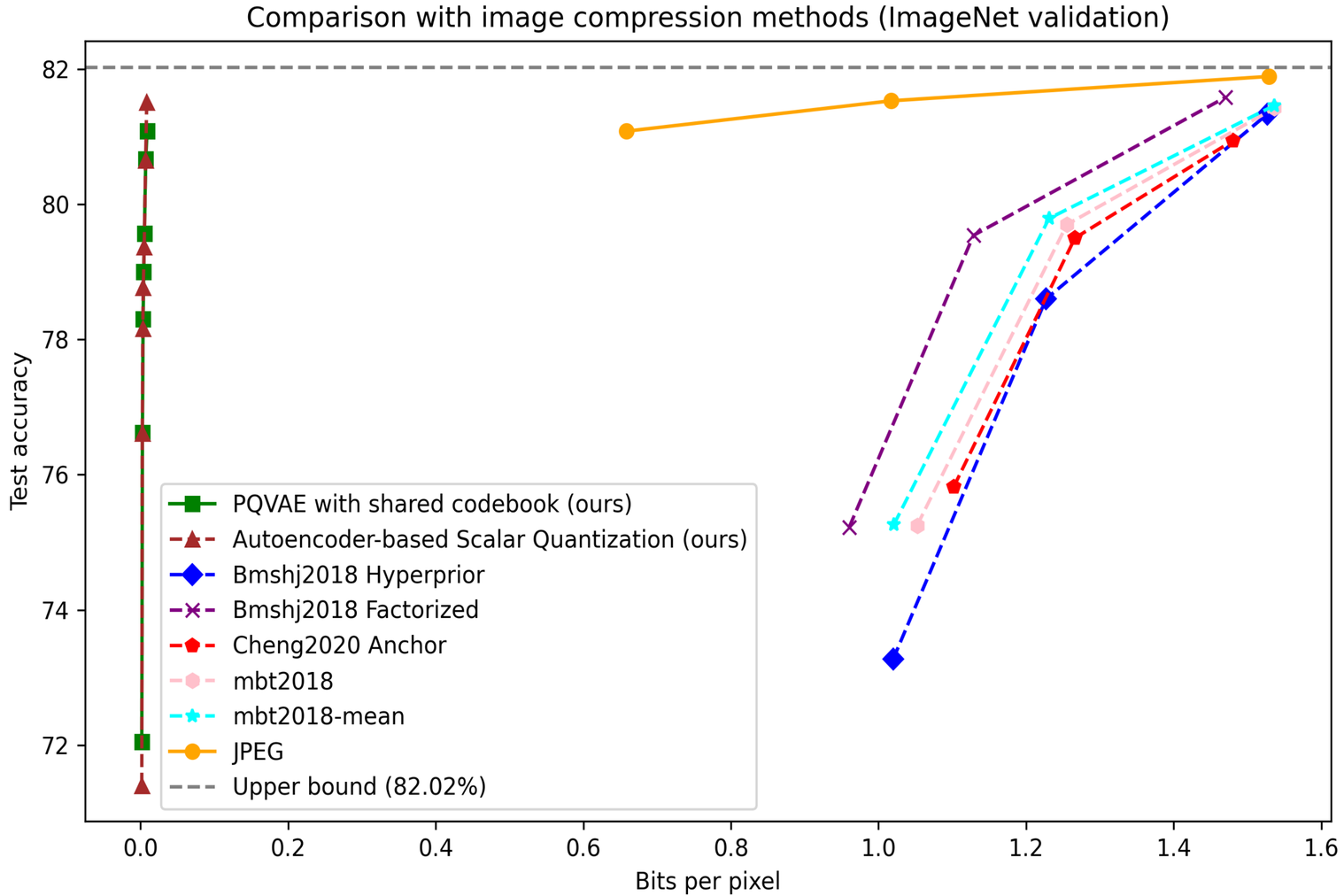
3.5 Comparison with image compression methods

Metrics:

- Test accuracy of image classification (same as before)
- bpp (bit per pixel)

Results:

- Our CLIP feature compression methods outperform far better than all image compression methods
- JEPG perform better than deep-learning-based image compression methods in that the pre-trained deep-learning-based image compression models might not align well with CLIP

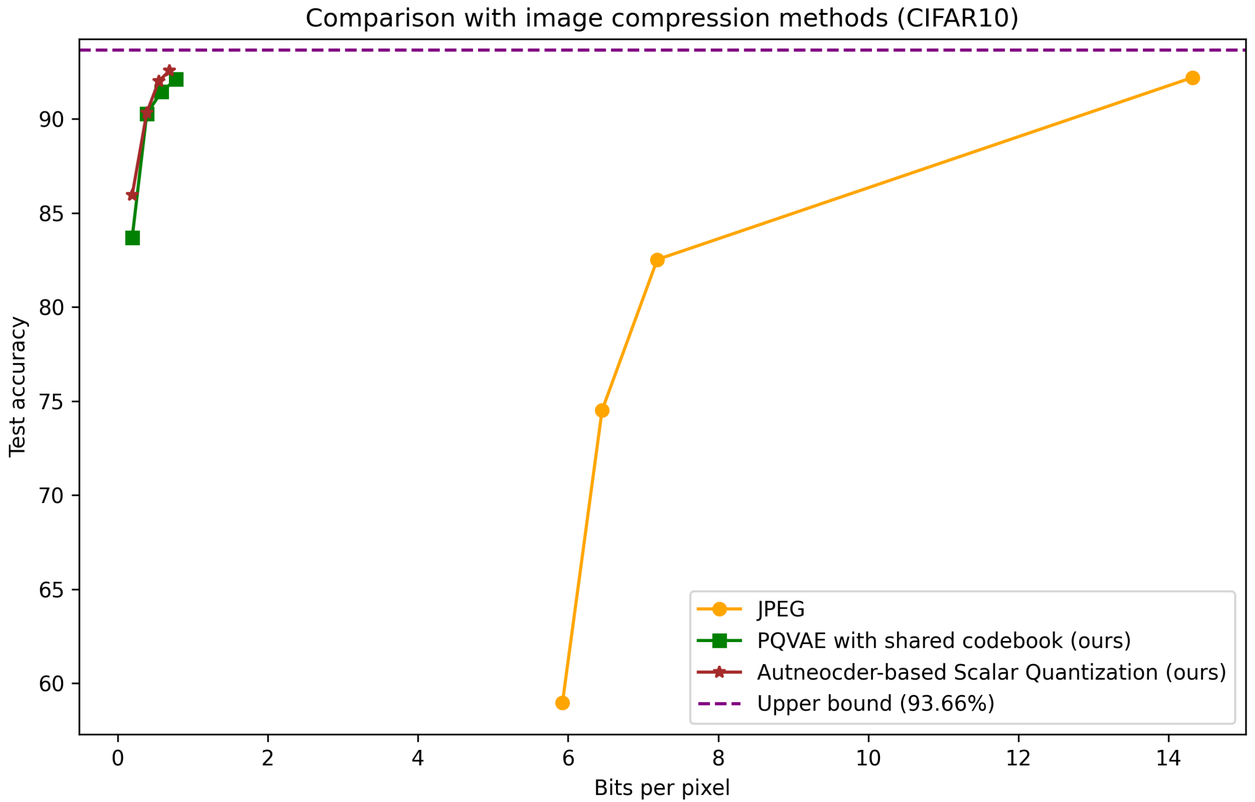


3. Evaluation

3.5 Comparison with image compression methods

- Application on imageset with small resolutions (CIFAR10: 32*32)
- Chen (2023) found the vulnerability of CLIP to JPEG compression for small resolution images: higher JPEG compression level leads to poorer performance on CLIP
- For most deep-learning-based image compression methods, compression of small resolution images is even impossible (≥64*64)
- bmshj2018 – factorized:

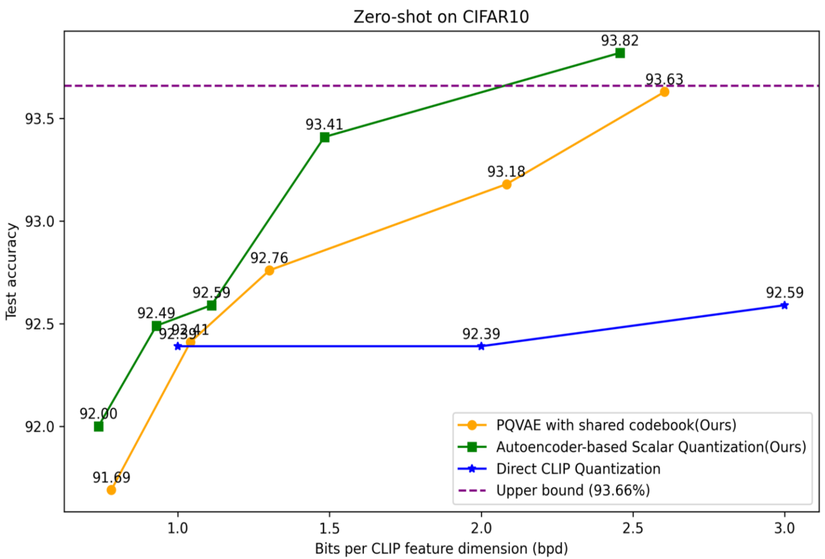
bpp	14.253666	15.610680	17.267
Test accuracy	9.86	9.96	9.97

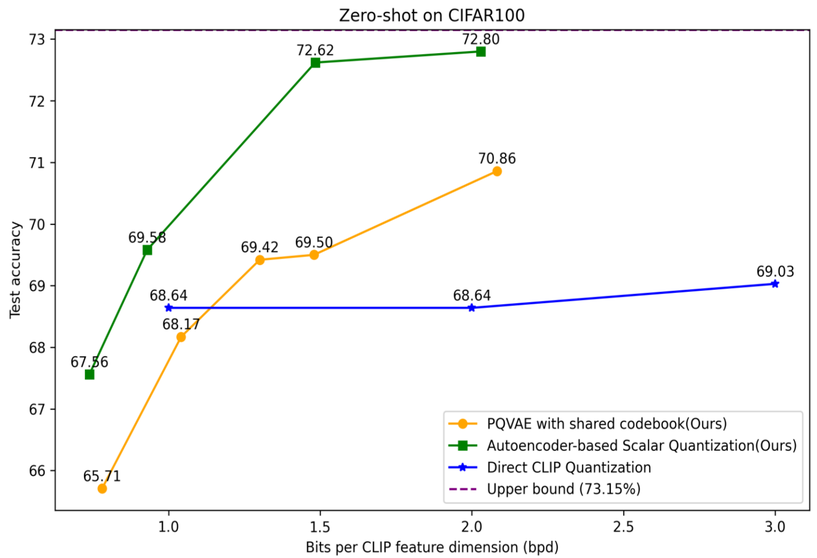


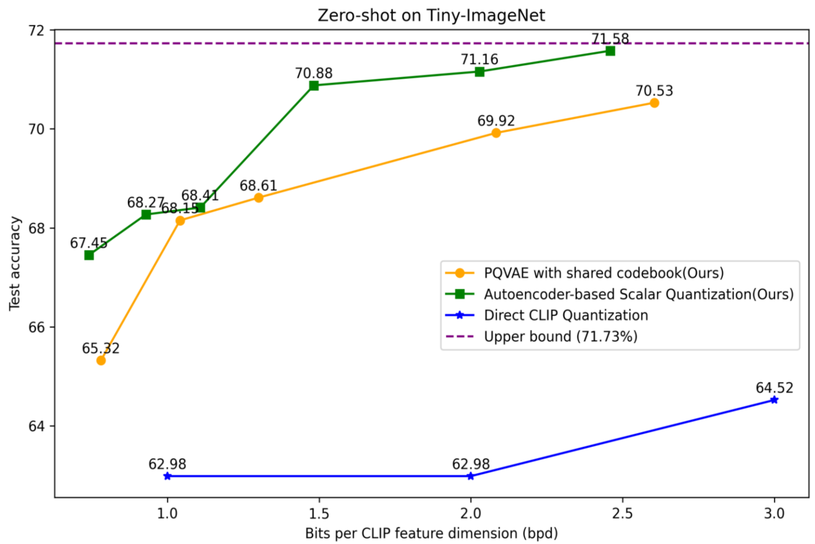
- C. Chen, V. P. Namboodiri, and J. Padget, “Understanding the vulnerability of clip to image compression,” arXiv preprint arXiv:2311.14029, 2023.

3. Evaluation

3.6 Zero-shot performance on other datasets







- Across all datasets, both methods demonstrate outstanding zero-shot performance, surpassing that of direct scalar quantization on CLIP features

- As the bpd constraint increases, both performance progressively approach the upper bound of the respective dataset.

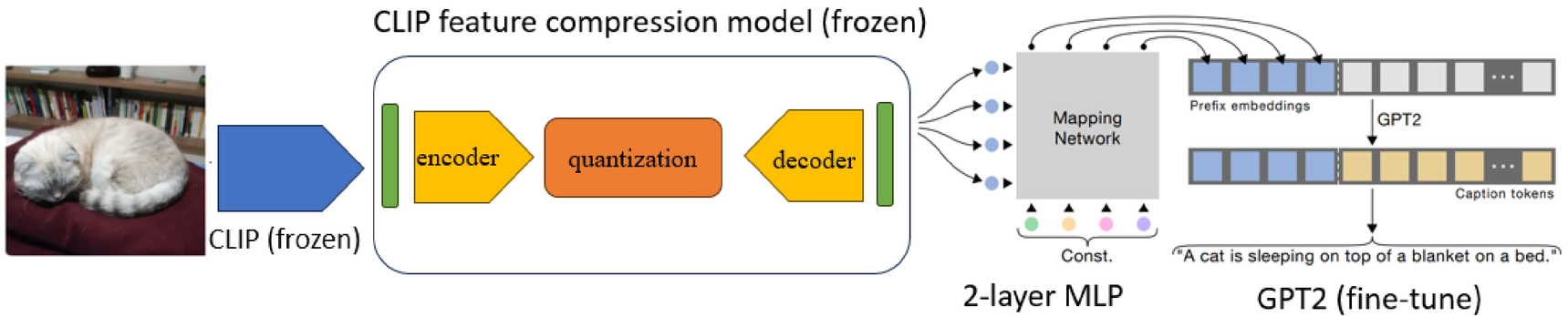
3. Evaluation

3.7 Performance of compressed CLIP features on downstream tasks

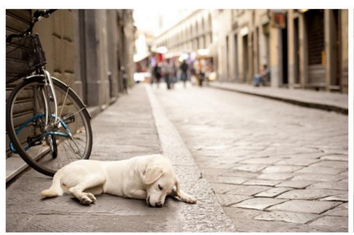
3.7.1 Image captioning

Settings of downstream tasks:

- In a real digital transmission scenario, the user's command (image/text) is first extracted by CLIP, then compressed and transmitted to the data center for task execution (over a noiseless channel).



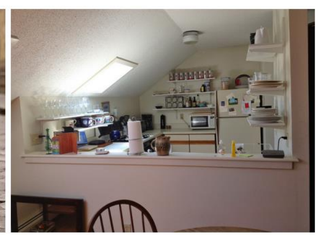
- CLIP and Compression model are frozen;
- Train the mapping network of MLP
- Fine-tuning GPT2



GT: A picture of a dog laying on the ground.

CLIP: A dog laying on the sidewalk next to a bicycle.

Compressed: A dog laying on the ground next to a bike.



A small kitchen with low a ceiling.

A kitchen with a stove, sink, and a refrigerator.

A kitchen with a stove, sink, and refrigerator.

There is small bus with several people standing next to it.

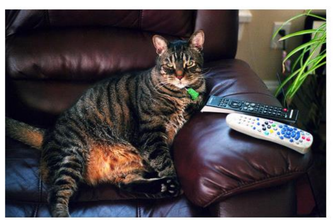
A truck driving down a street next to a park.

A crowd of people standing around a food truck.

GT: A woman in a trench coat playing with her kite.

CLIP: A woman flying a kite in a park.

Compressed: A woman standing in a park with kites flying in the air.



A cat on a leather chair next to remotes.

A cat laying on top of a couch next to a remote control.

A cat is laying on a couch with a remote control.

The clock tower is in the center of the building.

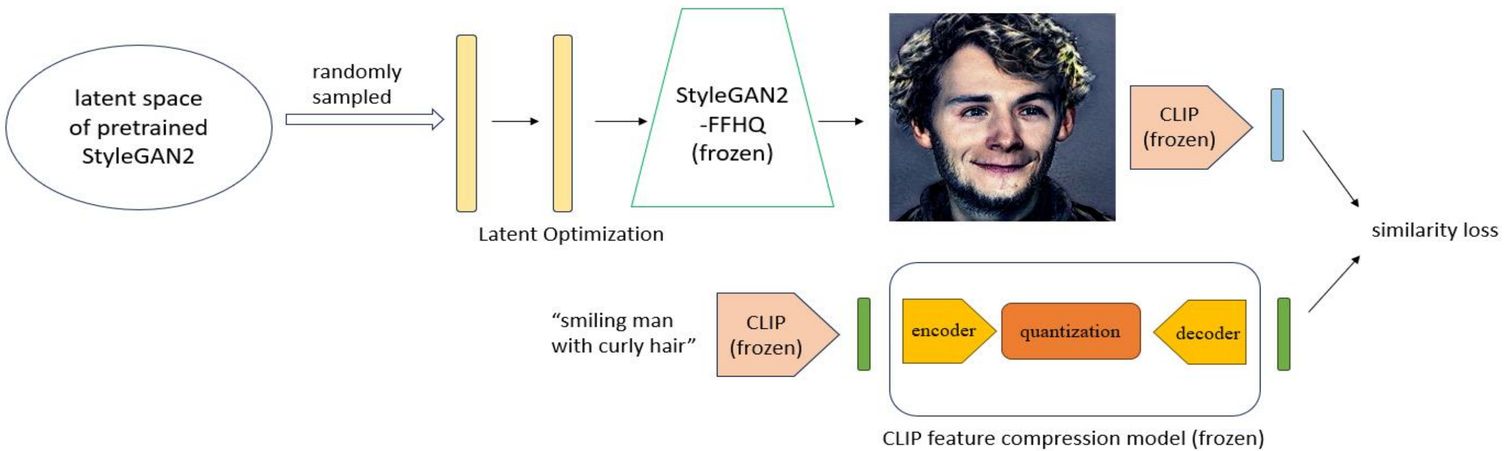
A clock tower on a building with a clock on each of it's sides.

A large clock tower in a city with a sky background.

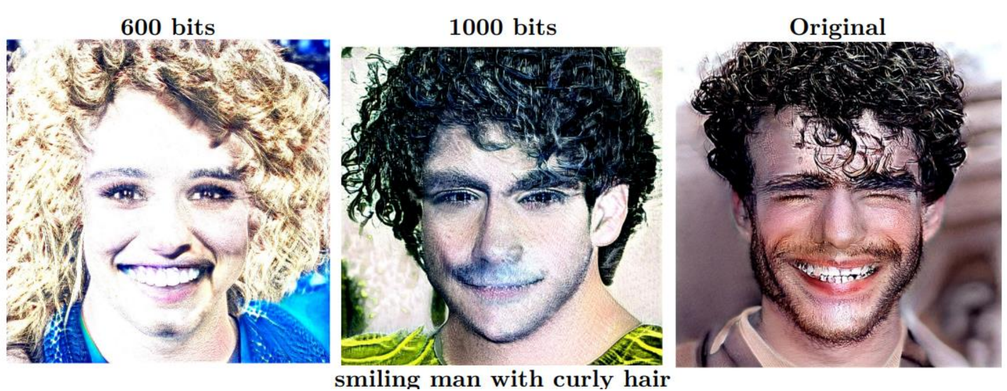
3. Evaluation

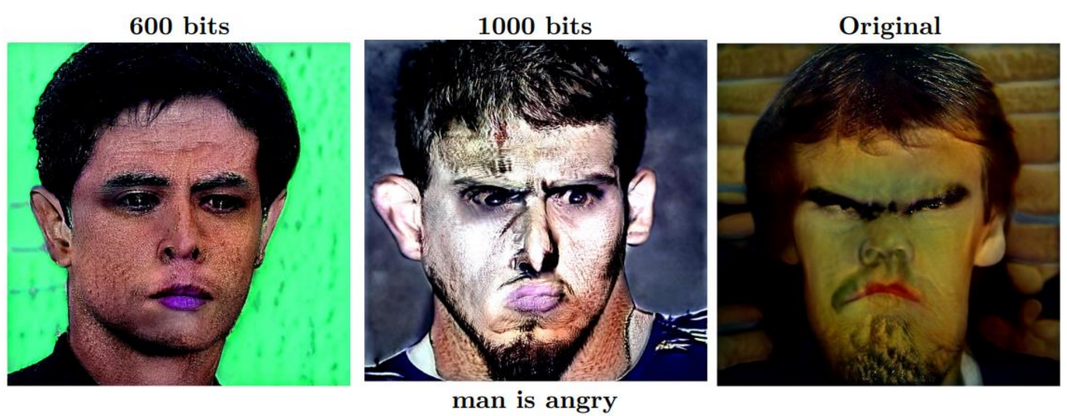
3.7 Performance of compressed CLIP features on downstream tasks

3.7.2 Image generation



- Optimize the latent representation, enabling it to stay close to the text prompt.
- All models are frozen.

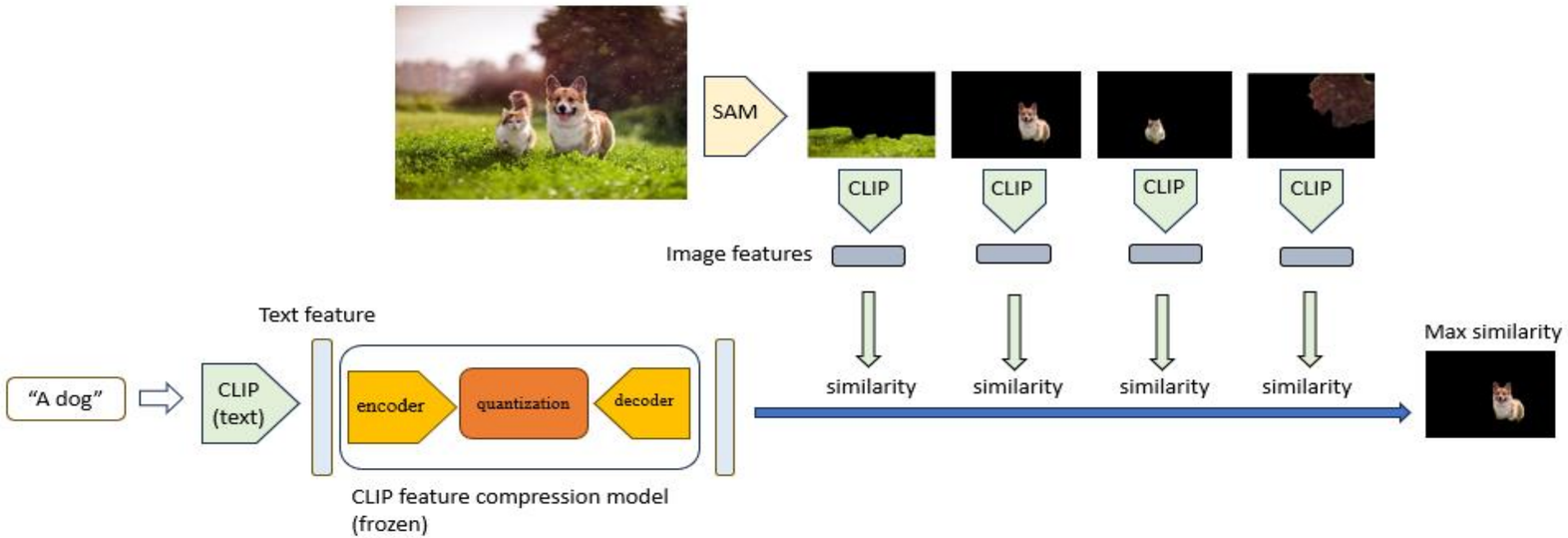




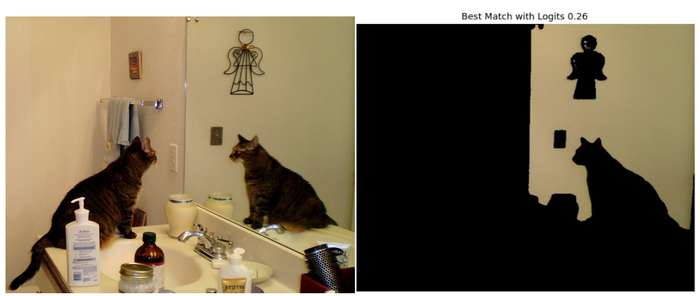
3. Evaluation

3.7 Performance of compressed CLIP features on downstream tasks

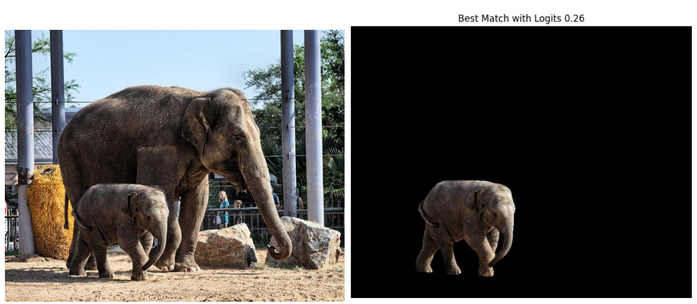
3.7.3 Targeted Object Segmentation



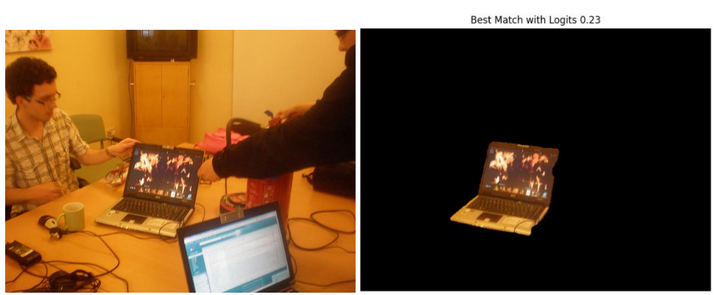
- No training / optimization process at all.
- Direct inference



cat in the mirror



the smaller elephant



laptop which people touch

Conclusion

- We present, for the first time, a novel approach to compressing CLIP features that aims to preserve semantics well under extreme compression (using the fewest bits).

- We propose two specific CLIP feature compression methods: PQ-VAE with shared codebook and Autoencoder-based scalar quantization. Each has its unique strength, and both effectively retain the semantic integrity of CLIP features, even under extremely high compression.

- We benchmark our compression methods against competitive image compression techniques, demonstrating that, while using significantly fewer bits, our methods preserve the semantic content of images equally well, if not better. Our method can also be applied to images of varying resolutions.

- We demonstrate the effectiveness of our compression methods' zero-shot performance across multiple datasets. Additionally, we show that the compressed CLIP features can still perform well on a wide range of downstream tasks with minimal impact on the performance, which include image captioning, image generation and targeted object segmentation.