

Learning to Structure an Image with Few Colors and Beyond

Yunzhong Hou, Liang Zheng*, and Stephen Gould

Abstract—Color and structure are the two pillars that combine to give an image its meaning. Interested in critical structures for neural network recognition, we isolate the influence of colors by limiting the color space to just a few bits, and find structures that enable network recognition under such constraints. To this end, we propose a color quantization network, ColorCNN, which learns to structure an image in limited color spaces by minimizing the classification loss. Building upon the architecture and insights of ColorCNN, we introduce ColorCNN+, which supports multiple color space size configurations, and addresses the previous issues of poor recognition accuracy and undesirable visual fidelity under large color spaces. Via a novel imitation learning approach, ColorCNN+ learns to cluster colors like traditional color quantization methods. This reduces overfitting and helps both visual fidelity and recognition accuracy under large color spaces. Experiments verify that ColorCNN+ achieves very competitive results under most circumstances, preserving both key structures for network recognition and visual fidelity with accurate colors. We further discuss differences between key structures and accurate colors, and their specific contributions to network recognition. For potential applications, we show that ColorCNNs can be used as image compression methods for network recognition.

Index Terms—Color quantization, explainable AI, image compression, deep clustering.

1 INTRODUCTION

IMAGES use both color and structure to convey information. Curious about the critical structures for neural network recognition, we isolate the contribution of colors to the recognition task, and find key structures that alone enable network recognition under very small color spaces. As a combination of shapes, textures, and other visual cues, the underlying structure emerges from the arrangement of different colors. Particularly, structures are only well presented when there exists a sufficient set of colors. As such, we investigate the interplay between colors and structures to help our final goal.

In the literature, a closely related line of work on the interplay between colors and structures is color quantization. Color quantization focuses on generating visually accurate images in restricted color spaces [1], [3]. This problem is *human-centered*, as it usually prioritizes the visual fidelity or quality for human viewing. To this end, traditional color quantization methods minimize the color distortion during quantization. Specifically, they cluster the pixels according to their RGB color values: pixels of similar colors are grouped into the same cluster, and pixels of dis-similar colors are assigned into different clusters. The color quantization process is finalized by assigning an overall color to all pixels for each cluster. In this manner, pixels in the color

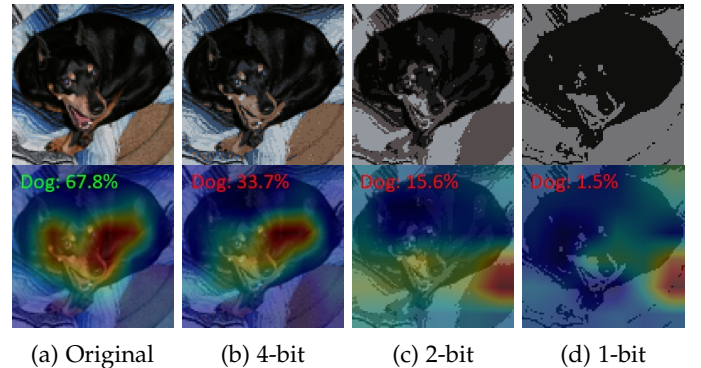


Fig. 1: Original and MedianCut [1] color quantized images (top) and corresponding class activation maps [2] with network confidence for the ground truth class (bottom). In “Original” (a), colors are described with 24 bits. When the color space reduces, the focus of the neural network deviates from the informative parts in (a) (correctly recognized), resulting in recognition failures in (b), (c), and (d).

quantized image have RGB color values very similar to the original ones, hence preserving visual fidelity. See Fig. 1 (b) for an example: in a 4-bit color space, the color quantization result is still visually similar to the original image in a 24-bit color space.

Finding key structures for network recognition is orthogonal to traditional color quantization problems, as it is *network-centered*: neural network recognition accuracy is its major focus, instead of human viewing experiences. To investigate the key structures for neural network recognition, we further reduce the color spaces to just a few bits, so as to minimize the contribution of colors to the recognition task. In this manner, we can ensure that the neural networks

- * Corresponding author.
- Yunzhong Hou, Liang Zheng, and Stephen Gould are with Research School of Computer Science, Australian National University, Canberra, ACT 2601, Australia. E-mail: {firstname.lastname}@anu.edu.au
- Yunzhong Hou, Liang Zheng, and Stephen Gould are with Australian Centre for Robotic Vision.

Manuscript received September 11, 2020; modified August 12, 2021; resubmitted-as-new January 15, 2022.

This work was supported by the ARC Discovery Early Career Researcher Award (DE200101283) and the ARC Discovery Project (DP210102801).

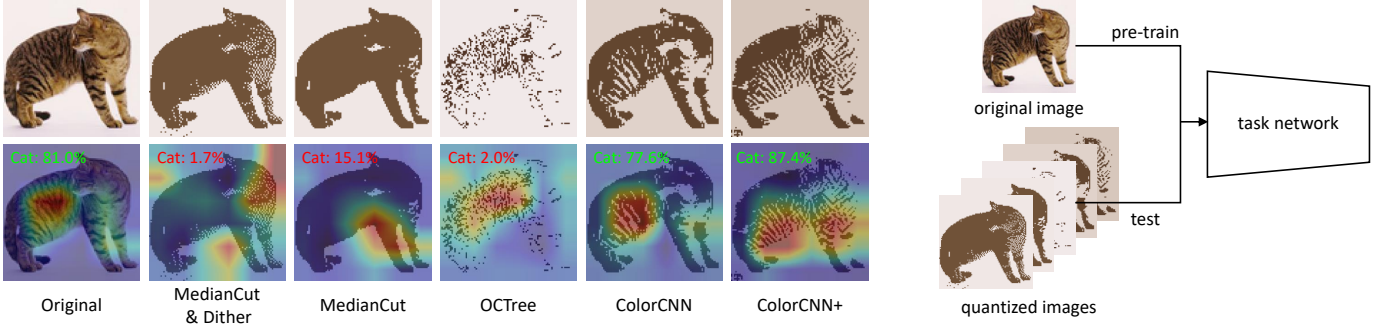


Fig. 2: **Quantization results:** “MedianCut & Dither” [1], [4], “MedianCut” [1], and “OCTree” [5] are traditional color quantization methods; “ColorCNN” and “ColorCNN+” are the proposed methods. **Evaluation:** We evaluate the color quantized images using a classification network pre-trained on the original images. **Comparison:** Traditional methods quantize the original image based on only RGB color values, and hence may lose important shapes and textures, resulting in recognition failures. On the other hand, through learning to structure, images quantized by our methods better preserve shapes and textures, and are successfully recognized by the pre-trained classifier.

primarily relies on the preserved structures for recognition. Natural images usually contain rich colors and structures, and further limiting the color space will inevitably compromise the structure. For example, in Fig. 1 (c) and (d), as we reduce the color space sizes all the way down to 1-bit (two colors), most structures vanish. When given such images in very limited color spaces, a neural network trained on original natural images might be ineffective, as there are neither accurate colors nor informative structures in such images. In this example, with the gradual reduction of the color space, the network fails to find the dog head and then the dog body, leading to recognition failures.

In this work, to identify the key structures, we design a color quantization method, ColorCNN, which learns to preserve informative structure within an image in an end-to-end manner. Through minimizing the classifier loss and improving the recognition accuracy, ColorCNN learns the informative structures for recognition under limited color space. Unlike traditional color quantization methods that rely on RGB color values to preserve accurate colors and visual fidelity for human viewing, ColorCNN exploits semantics to identify and preserve the critical structures for machine perception. As shown in Fig. 2, we evaluate the quantized images with a classifier pre-trained on original images. Images quantized by ColorCNN include richer structures like the overall shape, tabby stripes, and the cat head, enabling the classifier to successfully recognize the cat even under extremely limited color spaces.

Color quantization results from ColorCNN (Fig. 3 mid) show a clear preference towards *key structures* for network recognition over *visual fidelity* with accurate colors, which is quite different from the traditional approaches like MedianCut [1] (Fig. 3 top). Under small color spaces, *i.e.*, 1-bit or 2-bit, ColorCNN identifies the outline, logo, windshield, and wheels as key structures, and hence, allows for successful recognition. However, when the color space size increases, directly optimizing for accuracy can lead to overfitting, and no longer helps the vanilla ColorCNN to achieve competitive accuracy. The identified critical structures also provide limited help in such scenarios, as structures can be naturally supported by the large number of colors. To address this

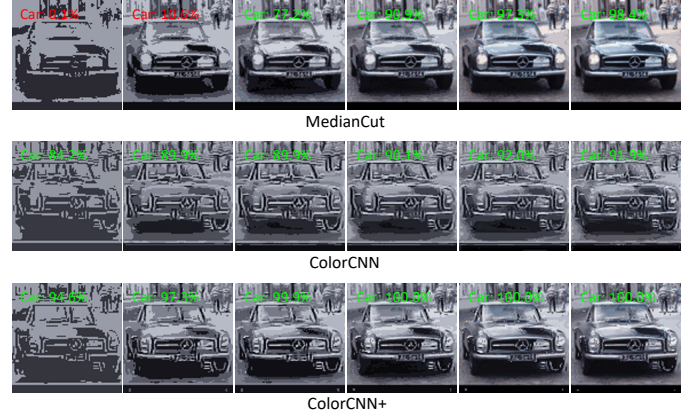


Fig. 3: Color quantization using traditional clustering-based method (MedianCut [1]), ColorCNN [6], and ColorCNN+. From left to right are results under 1 to 6-bit color spaces. In **small** (1 and 2-bit) color spaces, ColorCNN and ColorCNN+ maintain *key structures*, enabling network recognition. In **large** (5 and 6-bit) color spaces, optimizing directly for classification loss can cause overfitting, and no longer leads to competitive recognition accuracy for ColorCNN. ColorCNN+, on the other hand, outputs more visually accurate results with smaller *color distortions* (MedianCut is the most visually accurate). This approach addresses the overfitting issue, and thus gives higher accuracy.

issue, we design a new architecture in this journal extension, ColorCNN+, that optimizes for not only classification loss, but also visual fidelity under large color spaces (Fig. 3 bottom). As it successfully deals with the training accuracy overfitting issue in vanilla ColorCNN, ColorCNN+ achieves higher recognition accuracy under large color spaces.

This paper extends our conference version [6] in several critical aspects. **First**, the vanilla ColorCNN has to be re-trained for every specific color space size, which brings difficulties to its deployment and real-world usage. In its enhanced version, ColorCNN+, we use a single model to perform color quantization under different color space sizes. **Second**, for larger color spaces, color quantization

results from ColorCNN display low visual fidelity with large color distortions, and have dis-satisfactory recognition accuracy. In ColorCNN+, we minimize the color distortion by introducing a novel imitation learning approach, which learns to cluster the colors like traditional color quantization methods. This addresses the previous overfitting issue, and leads to higher visual fidelity (more accurate colors compared to ColorCNN) and competitive recognition accuracy under large color spaces. **Third**, we include more experiments on other classification settings, including multi-label classification and classification on stylized images. The new experiments enable us to compare structure preferences of different networks on different tasks, and provide us deeper insights into how networks recognize an image. **Fourth**, we show and discuss how informative structures and accurate colors benefit neural network recognition. Specifically, we find that despite their collaboration in large color spaces (which naturally supports more structures), visual fidelity (accurate colors) for *human* viewing and key structures for *network* recognition might contradict each other in small color spaces.

We demonstrate the effectiveness of ColorCNN and ColorCNN+ quantization on image classification tasks. It is found that both methods outperform traditional ones by a large margin under small color spaces, while ColorCNN+ remains competitive under large color spaces. Such results verify that the preserved patterns from ColorCNNs indeed help network recognition in small color space, and can be considered as key structures for network recognition. For applications, ColorCNNs stand as competitive image compression methods that enable effective neural network recognition while enjoying very low bitrates.

2 RELATED WORK

Color quantization. Color quantization [3], [7], [8], [9], [10] shrinks the color space sizes by grouping similar colors together and represent them with a new color. In this manner, color quantization can reduce image storage while keeping visual similarity to the original image. To best preserve visual fidelity, color quantization is usually formulated as a color value clustering problem. Many efficient color clustering methods are introduced, including the popular MedianCut [1] and OTree [5] algorithms. Dynamic programming [10] and peer group filtering [7] are also investigated for color quantization. Dithering [4], which removes visual artifacts by adding a noise pattern, is an optional step for better human viewing experience. Color quantization techniques are also applied in segmentation, including JSEG [9], SLIC [8], and more.

Human-centered image compression. Based on heuristics, many image compression methods are designed for human viewers. These methods fall into two categories, lossless compression, *e.g.*, PNG [11], and lossy compression, *e.g.*, JPEG [12], [13]. Color quantization also falls in the category of lossy compression. Its results, however, can be encoded in a lossless manner. The color quantized images can be represented as *indexed color* [14], and encoded with Portable Network Graphics (PNG) [11].

Recently, deep learning methods are introduced to image compression problems. Both recurrent methods [15], [16],

[17] and convolutional methods [18], [19], [20], [21] are investigated. Some method [16], [17] minimize distortion loss under different compression ratio. On the other hand, aiming to achieve a better compression ratio under multiple bitrate settings, some [22], [23] directly optimize rate-distortion instead. One possible drawback of these deep methods is that they have a much higher computation cost and need a separate decoder neural network.

Network-centered image compression. Traditional or deep-learning based, the aforementioned image compression methods are human-centered. However, in many cases, human-centered methods are not the best choice for neural network tasks. Liu *et al.* [24] points out that for segmentation, human-centered compression is not the best choice for 3D medical images. For 2D map data and 3D scene models, network-centered compression methods are designed for localization [25], [26]. Researchers use end-to-end trainable auto-encoder architecture for the map data compression.

Deep clustering. Using neural networks to solve the clustering problem is non-trivial, and stands as a challenging problem itself. Self-organizing maps [27] reduce the input dimension by creating a discretized grid as the representation of training samples. More recently, many investigate deep neural networks as a means for dimensionality reduction or representation learning. Many *clustering losses*, including k-means loss [28] (distance with k-means cluster center), cluster assignment hardening loss [29], [30], [31] (promotes more confident cluster assignments), cluster classification loss [32], [33] (deems the clusters as classes). For the final *cluster updates*, some adopt the established algorithms like k-means or Agglomerative clustering [30], [32], [34], [35], while others use neural networks to output cluster centers [28], [29], [31], [36] or soft cluster assignments [33], [37]. In this work, in order to preserve visual fidelity with color clustering, we investigate an alternative approach in deep clustering. Specifically, we adopt a novel imitation learning loss as the *clustering loss*, and a fully convolutional pipeline for the final *cluster update*.

3 METHODOLOGY

In this section, we first formulate the learning-to-structure problem mathematically in Section 3.1. Next, to identify and preserve the critical structures in original images, we design ColorCNN architecture and an end-to-end training method (see Fig. 4) in Section 3.2 and Section 3.3, respectively. Preliminary experiments find that ColorCNN effectively preserve key structures and enable network recognition under small color spaces, but overfits under large color spaces, suffering from poor visual fidelity and undesirable recognition accuracy. Identifying such problems in the vanilla ColorCNN design, we then introduce its improved version, ColorCNN+, which can preserve both key structures under small color spaces and visual fidelity (accurate colors) under large color spaces. We introduce the ColorCNN+ network architecture in Section 3.4, and the updated training pipeline in Section 3.5 and Section 3.6. ColorCNN+ also enables us to investigate how key structures and accurate colors contribute to neural recognition, which we take a deeper dive into in the next section.

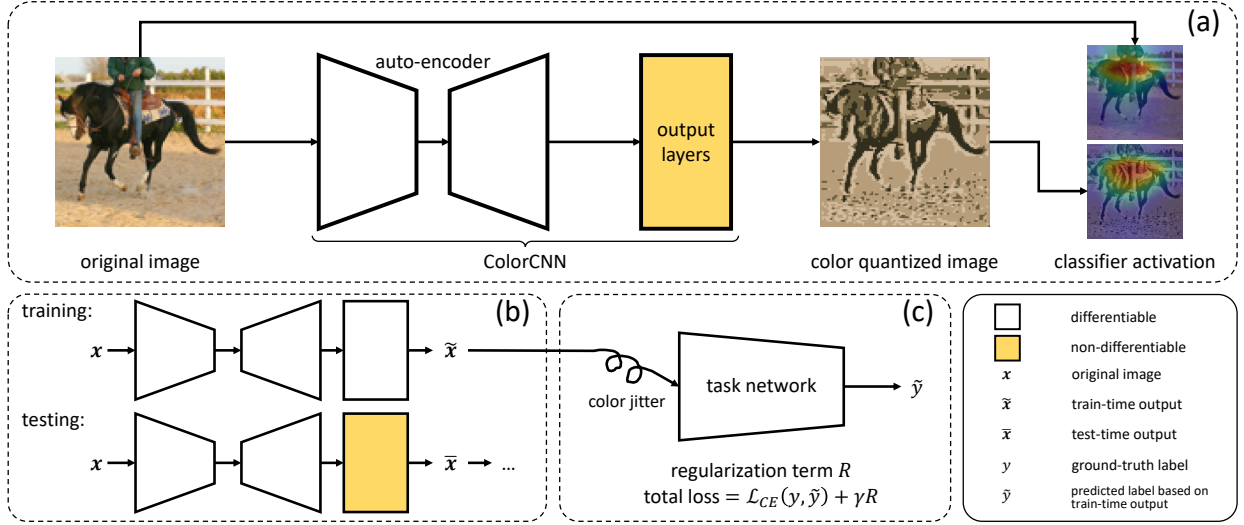


Fig. 4: Overview of the ColorCNN approach in finding key structures for network recognition. (a): ColorCNN applies network-centered color quantization in limited color space, so as to isolate the contribution of colors to the recognition task. (b): We replace the non-differentiable parts with approximations during training. (c): The ColorCNN network is trained with classification loss in an end-to-end manner. Regularization term R and color jitter are introduced to keep the approximation similar to the original network and prevent overfitting.

3.1 Problem Formulation

Given an input original image x and a color space size C , color quantization methods output the color quantized image $\bar{x}$, which can be represented and encoded by the color index map $M(x)$ and the color palette $T(x)$: using the color index map as a lookup table and filling in colors from the color palette, a color quantized image can be reconstructed.

Our objective then is to construct an image out of a few colors such that the color quantization result can still be correctly recognized by a pre-trained classifier. We consider the following,

$$\mathcal{L} = \mathcal{L}_{CE}(y, \tilde{y}) + \gamma R, \quad (1)$$

as our overall loss function, where $\mathcal{L}_{CE}(\cdot, \cdot)$ denotes the cross-entropy classification loss. The variable y denotes the ground truth label for image x , and $\tilde{y}$ denotes the pre-trained classifier output from the color quantized image. R is a regularization term and γ denotes its weight.

3.2 ColorCNN Architecture

We show the ColorCNN architecture in Fig. 5. Its first component is an U-net [38] auto-encoder that identifies the critical and semantic-rich structures.

Secondly, we use depth-wise (1×1 kernel size) convolution layer to create a softmax probability map of each pixel taking one specific color. This results in a C -channel probability map $m(x)$ (softmax over C -channel).

Then, for each input image x , the 1-channel color index map $M(x)$ is computed as the arg max over the C -channel probability map $m(x)$,

$$M(x) = \arg \max_c m(x). \quad (2)$$

The RGB color palette, $T(x)$, which is of shape $C \times 3$, is computed as average of all pixels that falls into certain quantized color index,

$$[T(x)]_c = \frac{\sum_{(u,v)} [x]_{u,v} \cdot \mathbb{I}([M(x)]_{u,v} = c)}{\sum_{(u,v)} \mathbb{I}([M(x)]_{u,v} = c)}, \quad (3)$$

where $[\cdot]_i$ denotes the i -th element or tensor of the enclosed entity, $\mathbb{I}(\cdot)$ is the identity function taking value 1 if its argument is true and 0 otherwise, and $\bullet$ denotes component-wise multiplication. For pixel (u, v) in a $W \times H$ image, $[x]_{u,v}$ denotes the pixel and its RGB value in the input image, and $[M(x)]_{u,v}$ represents its computed color index. $[T(x)]_c$ denotes the RGB value for the quantized color c .

Finally, the quantized image $\bar{x}$ is created via a table lookup session, which can be represented as,

$$\bar{x} = \sum_c [T(x)]_c \cdot \mathbb{I}(M(x) = c). \quad (4)$$

By combining Eq. 2, 3, 4, we finish the ColorCNN forward pass $\bar{x} = g(x)$ under color space size C .

3.3 End-to-End Learning

3.3.1 Differentiable Approximation

Fig. 6 shows the differentiable approximation used during training. To start with, we remove the arg max 1-channel color index map $M(x)$ in Eq. 2. Instead, we use the C -channel softmax probability map $m(x)$.

Next, we change the color palette design that follows. For each quantized color, instead of averaging from pixels of the same color index, we set its RGB color value $[t(x)]_c$ as the weighted average over all pixels,

$$[t(x)]_c = \frac{\sum_{(u,v)} [x]_{u,v} \cdot [m(x)]_{u,v,c}}{\sum_{(u,v)} [m(x)]_{u,v,c}}. \quad (5)$$

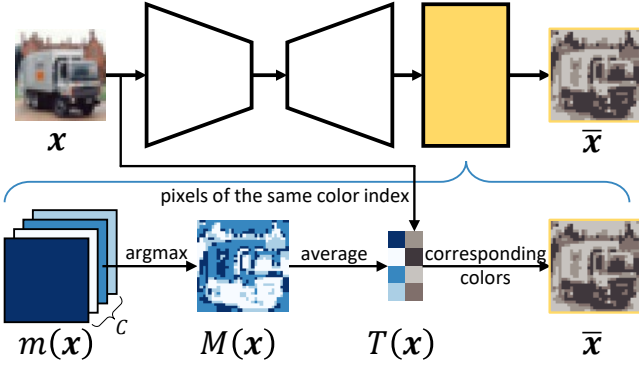


Fig. 5: ColorCNN architecture (**test-time**). First, the convolutional layers output a C -channel probability map $m(x)$ for C colors. Next, a 1-channel color index map $M(x)$ is created via the argmax function. Then, the color palette $T(x)$ is computed as average of all pixels that are of the *same color index*. At last, the color quantized image $\bar{x}$ is created via a *table look-up* session.

Here, the C -channel probability distribution $[m(x)]_{u,v}$ for a certain pixel (u, v) is used as the contribution ratio of that pixel to all C colors. This will result in a slightly different color palette $t(x)$.

Lastly, we change the table look-up process from the original forward pass into a weighted sum. For quantized color with index c , we use $[m(x)]_c$ as the intensity of expression over entire image. Mathematically, the train-time quantized image $\tilde{x}$ is computed as,

$$\tilde{x} = \sum_c [t(x)]_c \cdot [m(x)]_c. \quad (6)$$

By combining Eq. 5, 6, the forward pass for ColorCNN during training can be formulated as $\tilde{x} = \tilde{g}(x)$. At last, we substitute $g(\cdot)$ with $\tilde{g}(\cdot)$ in Eq. 1 for end-to-end training.

3.3.2 Overfitting Prevention

The two forward pass $g(\cdot)$ and $\tilde{g}(\cdot)$ behave very differently. See Fig. 7 for a side-by-side comparison of their outputs. The test-time output $\bar{x}$ only has C colors, whereas the train-time output $\tilde{x}$ has more than C colors. The main reason for this mismatch boils down to the difference between hard (test-time) and soft (train-time) assignments. As shown in Fig. 5, the one-hot approach allows influence only from *some* pixels to one quantized color, and from *one* color to any quantized pixel. On the other hand, in Fig. 6, with softmax function, *all* pixels influence all colors in the palette, and *all* colors in the palette contribute to each pixel in the output image.

Trained with the approximation $\tilde{g}(\cdot)$ instead of the test-time model, the proposed ColorCNN encounters overfitting. However, more freedom leads to easy convergence during the training, while the test-time results might still be struggling. In the following paragraphs, we introduce regularization terms and data augmentation as means to combat such overfitting.

Regularization. ColorCNN only uses image classification loss as supervision. As such, there is no guarantee that each image contains C colors. For this problem, we

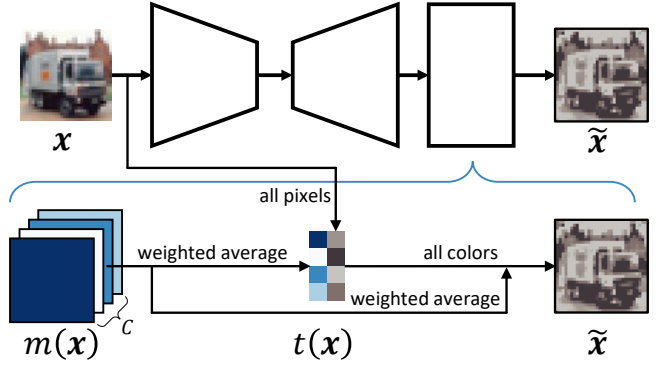


Fig. 6: The differentiable approximation (**train-time**). The C -channel probability map $m(x)$ is used instead of the argmax color index map $M(x)$. Next, the color palette $t(x)$ is adjusted as weighted average over *all* pixels. At last, instead of table look-up, the quantized image $\tilde{x}$ is computed as the weighted average of *all* colors in the color palette.

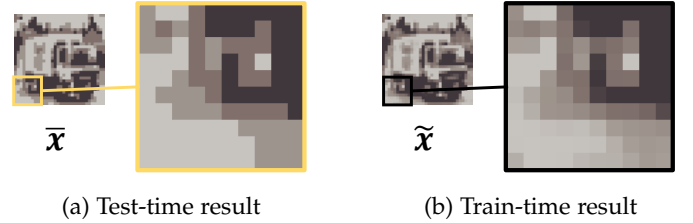


Fig. 7: Comparison between test-time result $\bar{x}$ and train-time result $\tilde{x}$. Each pixel in $\tilde{x}$ is a weighted average of all colors in its palette. Thus, more colors are introduced.

propose a color appearance regularization term R_{color} that encourages all C colors to be selected by at least one pixel in an image. For all pixels in an image, we take the maximum probability value along the color channel $c \in \{1, \dots, C\}$, and use this value as our regularization term

$$R = R_{\text{color}} = -\frac{1}{C} \times \sum_c \max_{(u,v)} [m(x)]_{u,v}. \quad (7)$$

Color jitter. In order to prevent overfitting, during training, we add a jitter $\xi \times n$ to the color quantized image $\tilde{x}$ after normalization as a form of data augmentation. The noise n is sampled from a Gaussian distribution $\mathcal{N}(0, 1)$. ξ denotes its weight. This creates a more difficult train-time output for the classifier to recognize, and helps to reduce overfitting.

The overall approach for ColorCNN is shown in Fig. 4.

3.4 ColorCNN+ Architecture

ColorCNN helps us analyze the critical structures for neural network recognition and specializes in small color spaces. However, it has two problems. **First**, ColorCNN architecture only supports one color space size, making it difficult to deploy. **Second**, ColorCNN overfits in larger color spaces (see Fig. 3), leading to less visually accurate outputs (for human viewing) and dis-satisfactory recognition accuracy (for machine perception). In this section, on top of ColorCNN, we build a novel color quantization architecture,

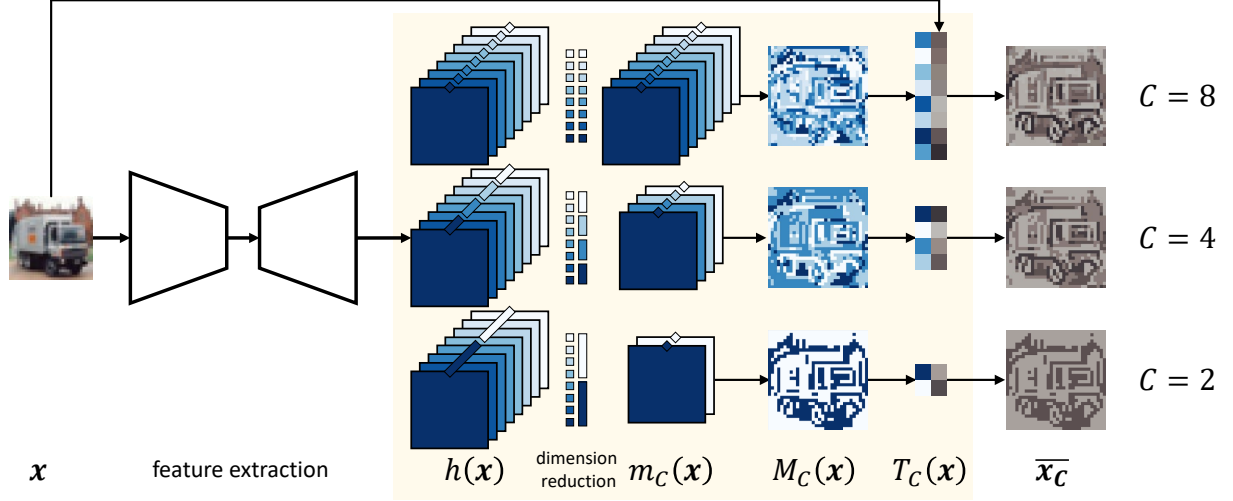


Fig. 8: ColorCNN+ supports multiple color space sizes in one model. Different from ColorCNN where the color space size C is used to decide the network structure (feature extractor output channel), ColorCNN+ has a fixed structure with auto-encoder output channels of D , where D is a predefined number. For multiple color space size support, we reduce the feature map dimension from D to C via channel-wise average pooling to formulate color probability maps $m_C(x)$. The rest of the ColorCNN+ including index map $M_C(x)$ and color palette $T_C(x)$ follows the same design as ColorCNN.

ColorCNN+, that supports multiple color space size settings in one model, while also capable of outputting results with high visual fidelity and minimal color distortions via deep clustering. Such improvements allows for not only easier real-world application, but also more discussion and deeper insight on informative structures and visual fidelity (accurate colors).

ColorCNN+ builds on the architecture of ColorCNN. An illustration of the ColorCNN+ architecture is shown in Fig. 8. Like ColorCNN, it still adopts a U-net auto-encoder as its feature extractor. After that, rather than directly outputting a C -channel color probability map as in vanilla ColorCNN, in its enhanced version, we **first** introduce a lower dimension *bottleneck layer*, so as to limit the excessive information and better combat the overfitting issue. **Next**, this bottleneck layer is followed by a D -channel feature map $h(x)$, where D is a relatively large (larger than color space sizes in our experiment) fixed number. This D -channel feature map $h(x)$ serves as building blocks of the color probability map, and we can **then** create different color probability maps $m_C(x)$ for different color space sizes C by reducing the dimension from the fixed number D to the color space size C . This dimension reduction is achieved via channel-wise average pooling,

$$[h_C(x)]_c = \left\lfloor \frac{D}{C} \right\rfloor \sum_{d=\lfloor \frac{c-1}{C} \times D \rfloor}^{\lfloor \frac{c}{C} \times D \rfloor} [h(x)]_d, \quad (8)$$

where $h_C(x)$ is the dimension reduction result of channel-wise average pooling over the D -channel feature map $h(x)$.

Following that, we adopt a new design when creating the C -dimensional color probability map $m_C(x)$. In vanilla ColorCNN, during training, all pixels in the original image x contribute to the color palette $t_C(x)$ and quantized image $\tilde{x}_C$, as long as the color probability map $m(x)$ is non-zero. Compared to the test-time pipeline where only the

pixels belonging to a certain quantized color contributes to the color palette and the final output, the training pipeline has significantly higher degree of freedom and is prone to overfitting. As such, in ColorCNN+, we modify our design of color probability map $m_C(x)$ by allowing up to *top- K nonzero terms*: at each pixel, $m_C(x)$ is assigned with the softmax probability $\sigma(\cdot)$ over K elements for channels that have top- K elements of $h_C(x)$, or 0 for other channels,

$$[m_C(x)]_c = \begin{cases} \sigma([h_C(x)]_c), & \text{if } c \in \text{top}_K[h_C(x)]_c, \\ 0, & \text{else.} \end{cases} \quad (9)$$

The rest of ColorCNN+ forward pass follows the original design in ColorCNN. To start with, we create the 1-channel color index map $M_C(x)$ with Eq. 2. The color palette $T_C(x)$ is then created via Eq. 3. At last, the C -color quantized image $\bar{x}_C$ is generated via Eq. 4. Combining all components, we have the forward pass of ColorCNN+ $\bar{x}_C = g(x, C)$.

3.5 Deep Clustering in ColorCNN+

In order to address the previous issue in ColorCNN where the larger color spaces are not effectively utilized (directly optimizing for train-time accuracy can lead to overfitting, see Fig. 3), ColorCNN+ learns from clustering-based traditional color quantization methods to maintain visual fidelity. In fact, we believe that promoting the higher visual quality also allows for easier network recognition under larger color space, as the large color spaces naturally allows for more structures, making it no longer the bottleneck for recognition accuracy. This deep clustering support also allows for more discussion how informative structures and accurate colors benefit neural network recognition.

3.5.1 Fully Convolutional Cluster Update

ColorCNN+ aims to preserve both key structures in small color spaces and accurate colors in large color spaces. It

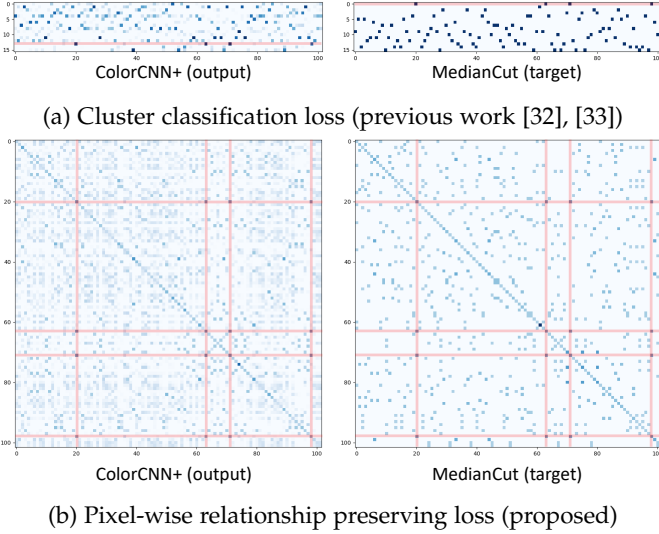


Fig. 9: Clustering loss computation for $C = 16$ quantized colors over $N = 102$ sampled pixels. Highlighted are the same group of pixels with different cluster indices given by different cluster updates.

is desired that the network can output an *overall* decision rather than having separate branches for decisions under different scenarios.

As for the clustering problem formulation, ColorCNN+ neither introduces a separate clustering step as [30], [32], [34], [35], [39], nor outputs the cluster centers [28], [29], [31], [36] and assign cluster indices according to feature distance. Instead, it follow the same per-pixel classification formulation as in vanilla ColorCNN: estimating a soft assignment for each pixel to one of C clusters. This creates a fully convolutional pipeline that does not need additional steps for clustering updates. Furthermore, we can easily incorporate the clustering results (for visual fidelity and accurate colors in large color spaces) into the current semantic-based results (for key structures in small color spaces) and make an *overall* decision with the color probability map $m_C(\mathbf{x})$.

3.5.2 Imitation Learning

For the ColorCNN+ architecture to learn color clustering, we introduce a new imitation learning process. Since we directly estimate the soft cluster assignment, the k-means loss [28] (requires a separate clustering step) and cluster assignment hardening loss [29], [30], [31] (when estimating cluster centers) from previous works are not suitable. This leaves us with the cluster classification loss [32], [33] (for soft cluster assignment estimation) as the only candidate in the literature, which treat the cluster index as class and formulate deep clustering as a cluster classification problem. However, this approach also has its own problem. Unlike classification problems where the classes are fixed, in clustering problems, the cluster indices are randomly generated and cluster indices from different cluster updates (ColorCNN+ as output or MedianCut as target) do not match: as shown in Fig. 9 (a), the two highlighted clusters, despite having different cluster indices, consist of the same pixels. This can cause large cluster classification loss, whereas in

fact minimal loss should come from these pixels as the two decisions are similar.

In this case, we investigate alternative choices that do not rely on the strict correspondence between cluster indices. Inspired by works in related fields that preserve batch-wise, pixel-wise, or channel-wise relationships on the *feature maps* [40], [41], [42], in this work, we compute the pixel-wise self correlations between *soft cluster assignments* $m_C(\mathbf{x})$ for clustering supervision. As shown in Fig. 9 (b), the pixel-wise relationship preserving loss only measures relationships between cluster assignments at each pixel. Unlike the cluster index, the pixel arrangements are always the same between different cluster updates, making it a more suitable supervision for deep clustering.

Given ColorCNN+ soft assignment $m_C(\mathbf{x})$ and MedianCut one-hot assignment $m_C^*(\mathbf{x})$, we **first** sample N pixels over $H \times W$ pixels to produce $\mathcal{M}, \mathcal{M}^* \in \mathbb{R}^{C \times N}$. **Next**, we compute the pixel-wise self correlations,

$$A = \mathcal{M} \cdot \mathcal{M}^T, \quad A^* = \mathcal{M}^* \cdot (\mathcal{M}^*)^T, \quad (10)$$

where $A, A^* \in \mathbb{R}^{N \times N}$. We **then** normalize them row-wise,

$$\tilde{A}_{[i,:]} = \frac{A_{[i,:]}}{\|A_{[i,:]\|_2}, \quad \tilde{A}^*_{[i,:]} = \frac{A^*_{[i,:]}}{\|A^*_{[i,:]\|_2}, \quad (11)$$

where $[i,:]$ indicates the i -th row in a matrix. **At last**, we define the relationship preserving loss as mean square error (MSE) between the normalized self correlation matrices,

$$\mathcal{L}_{RP} = \frac{1}{N} \|\tilde{A} - \tilde{A}^*\|_F^2, \quad (12)$$

where $\|\cdot\|_F$ denotes the Frobenius norm (entry-wise $\mathcal{L}_2$ norm for matrix).

Overall, we re-write the loss function in Eq. 1 as

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{RP} + \gamma R, \quad (13)$$

where λ is a hyper-parameter for the clustering loss weight.

3.6 Training Details in ColorCNN+

3.6.1 Regularization

In addition to the color appearance regularization term R_{color} for ColorCNN (see Eq. 7), for ColorCNN+, we introduce two new regularization terms. A information entropy regularization term R_{info} to encourage more evenly-distributed color choices; and a confidence regularization term R_{conf} to balance out R_{info} and increase confidence for the color probability map $m_C(\mathbf{x})$.

Information entropy as regularization. In order to maximize the information carried by the image so that the classifier can successfully recognise it, we aim to maximize the *information entropy* [43] of the color quantized image. For the color quantization task, we want the number of pixels that take each of the $c \in \{1, \dots, C\}$ quantized colors to be evenly-distributed,

$$R_{\text{info}} = -\mathcal{H} \left(\frac{1}{H \times W} \sum_{u,v} [m_C(\mathbf{x})]_{u,v} \right), \quad (14)$$

where $\mathcal{H}(\cdot)$ denotes the entropy function over the color channel $c \in \{1, \dots, C\}$.

Confidence as regularization. One side-effect of our information entropy regularization R_{info} is that it also makes the color probability map $m_C(\mathbf{x})$ less like one-hot. In light of this, we propose a confidence regularization term, which promotes the color probability map to be more confident at every pixel. We also use the information entropy function $\mathcal{H}(\cdot)$ to calculate this confidence regularization term,

$$R_{\text{conf}} = \frac{1}{H \times W} \sum_{u,v} \mathcal{H}([m_C(\mathbf{x})]_{u,v}). \quad (15)$$

This confidence regularization R_{conf} focuses on *pixel-wise* color distribution, whereas the information regularization R_{info} focuses on *image-wise* color distribution.

Combining all three components, we have the final regularization term for ColorCNN+,

$$R = R_{\text{color}} + \alpha R_{\text{info}} + \beta R_{\text{conf}}, \quad (16)$$

where α and β specify the ratio for the three components.

3.6.2 Data Augmentation

To better train ColorCNN+, we make some adjustments to the data augmentations. Most importantly, we move random cropping from before ColorCNN+ to after ColorCNN+. If applied before the quantization network, ColorCNN+ has to waste some of its network capacity on dealing with the black bars artifacts. When moved to after quantization network, such augmentation can still add diversity to the classifier training data and help to combat overfitting. In addition, we also introduce some more augmentation after the quantization network, in addition to the color jitter and random cropping, including random erasing [44], rotation, and horizontal flipping.

3.6.3 Task Selection

For each forward pass in training, we must specify a color space size $C \in \{1, \dots, 64\}$ (as we consider at most 64 colors in our experiments) for ColorCNN+. We refer to this selecting different color space size for ColorCNN+ training as task selection. Throughout the entire training process, we select different tasks (color space sizes) to give ColorCNN+ sufficient supervision for different color space sizes it could encounter during testing.

The pace of changing the task is a very important aspect of successful training of ColorCNN+. Similarly, Wu *et al.* [45] study the task selection problem for video understanding, where different tasks refer to different video resolutions and batch sizes. It is found that neither too fast nor too slow the pace in task changes lead to high performance. In our color space size selection task, we also witness a similar phenomenon, where changing task per-batch or per-epoch both lead to inferior performance. As such, we empirically choose an intermediate pace in task change for ColorCNN+ training at once every 20 batches.

4 EXPERIMENTS

4.1 Experimental Setup

Datasets. We evaluate on four single-label classification datasets, one multi-label classification dataset, and one stylized image dataset. Single-label classification is the default task if not specified.

TABLE 1: Classification accuracy (%) of different networks on different datasets.

	CIFAR10	CIFAR100	STL10	Tiny200
AlexNet [46]	86.9	62.1	73.6	50.9
VGG16 [47]	93.5	73.1	81.5	62.8
ResNet18 [48]	94.6	76.4	83.4	65.5

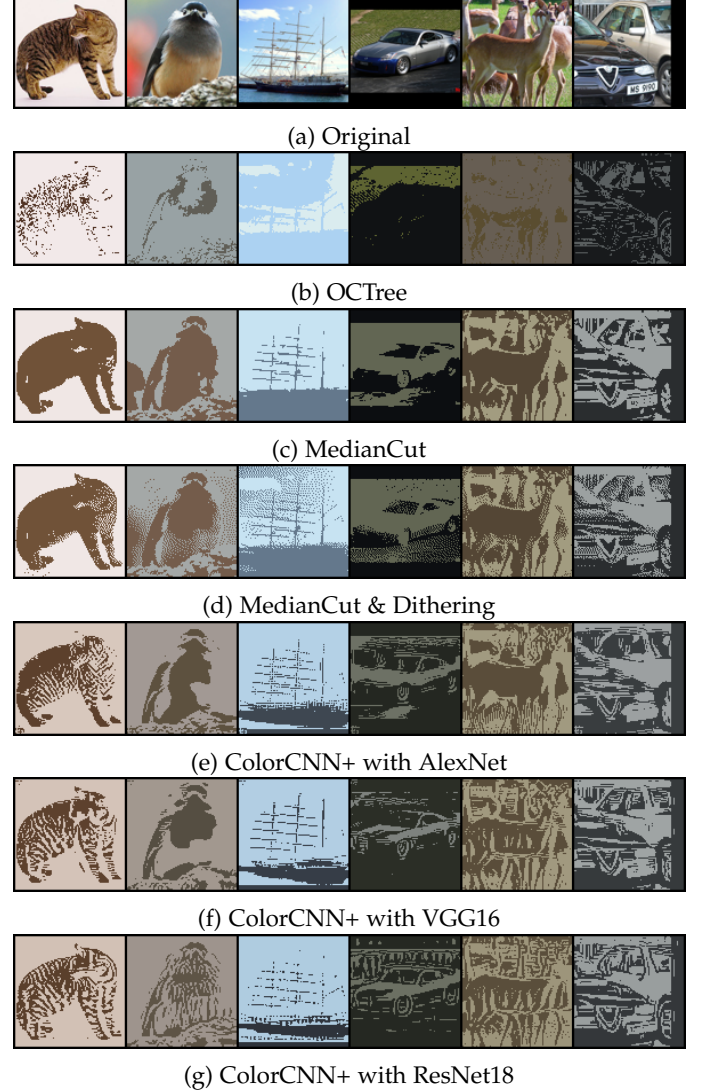


Fig. 10: 1-bit color quantization results. ColorCNN+ identifies both *textures* and *shapes* as critical structures for neural network recognition. Also, different classifiers prefer different patterns. For more discussion, please see Section 4.3.

For single-label classification, we use the following four datasets. CIFAR10 and CIFAR100 datasets [49] include 10 and 100 classes of general objects, respectively. STL10 dataset [50] contains 10 image classes but with a higher resolution. Tiny-imagenet-200 dataset [51] is a subset of ImageNet-1K dataset [52]. It has 200 classes of medium resolution images. We omitted experiments on the full ImageNet-1K dataset due to its high complexity.

For multi-label classification with multiple salient objects, we introduce Pascal VOC 2012 dataset [53].

Stylized ImageNet includes images in arbitrary artistic



Fig. 11: Color quantization results under different color space sizes. Column (a) shows the original images. Column (b)-(d), (e)-(g), and (h)-(j) show MedianCut [1], ColorCNN, and ColorCNN+ results for 1-bit, 3-bit, and 5-bit color spaces, respectively. ColorCNN prioritizes the informative structures over visual fidelity (accurate colors) on both small and large color spaces. In comparison, ColorCNN+ identifies and preserves the informative structures under small color spaces, and outputs visually accurate results under large color spaces. For more quantitative results, see Fig. 12.

styles [54]. We create a stylized dataset, Style14, that consists of 14 classes from the full ImageNet-1k dataset [52].

Classification networks. We choose AlexNet [46], VGG16 [47] and ResNet18 [48] for classification network. All classifier networks are trained for 60 epochs with a batch size of 128, except for STL10, where we set the batch size to 32. We use an SGD optimizer with a momentum of 0.9, L2-normalization of 5×10^{-4} . We choose the 1cycle learning rate scheduler [55] with a peak learning rate of 0.1. We report the classification accuracy on for single-label classification datasets in Table 1.

Multi-label classification uses binary cross entropy loss for training, and stylized image classification is trained in the same manner as single-label classification on natural images. We use ResNet18 architecture for both settings. Their performance can be found in Table 2 and Table 3.

ColorCNN. We train ColorCNN on top of the original image pre-trained classifier. We set the hyper-parameters as

follows. For the regularization and color jitter weight, we set $\gamma = 1$ and $\xi = 1$. We also normalize the quantized image by $4 \times$ the default variance of the original images, so as to enable easier training. The gradient descent optimizer runs 60 epochs with a batch size of 128. Similar to classification networks, we also reduce the batch size to 32 on the STL10 dataset. The SGD optimizer for ColorCNN training is the same as for classifier networks. For the learning rate scheduler, we choose Cosine-Warm-Restart [56] with a peak learning rate at 0.01, minimal learning rate at 0, and uniform restart period of 20 epochs.

ColorCNN+. We make the following modifications to ColorCNN+ over ColorCNN. Architecture-wise, we add a 16-dimensional bottleneck layer before the $D = 256$ -dimensional feature map, and set the number of non-zero terms in color probability map as $K = 4$. For the hyper-parameters, we set the weight for the clustering loss $\lambda = 3$, and the weights for regularization terms as $\alpha = 1$ and $\beta = 1$.

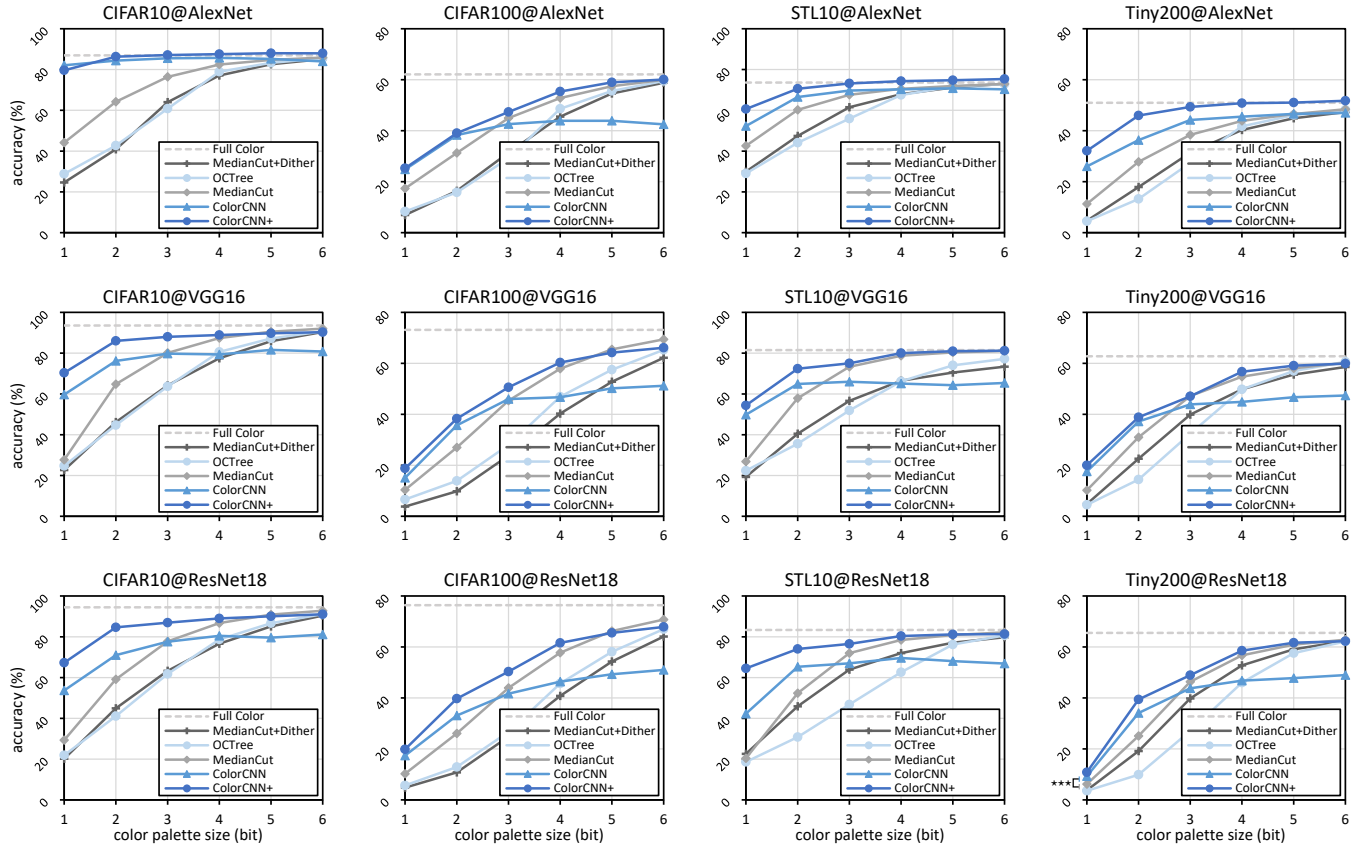


Fig. 12: Classification accuracy of color quantized images on four datasets with three networks. Compared to traditional methods, both ColorCNNs are significantly better under small color spaces, while ColorCNN+ remains competitive under large color spaces. *** means that the accuracy difference is **statistically very significant** (i.e., p -value < 0.001).

As for the number of pixels considered in our clustering loss, we set N as $0.3 \times$ the number of pixels for memory concerns. We train ColorCNN+ for 300 epochs and keep other hyperparameters the same.

All experiments are run on one RTX-3090 GPU.

4.2 Evaluation of ColorCNNs

We evaluate ColorCNNs and compare them against MedianCut [1], OCTree [5], and MedianCut with dithering [4].

4.2.1 Visualization

Visualization results in small color spaces. As the color space size shrinks, pixels struggle to support structures. Under such scenarios, keeping critical and informative structures becomes the key to successful network recognition. In a minimum 1-bit color space (Fig. 10 and Fig. 11), under the same setting, traditional color quantization method like MedianCut merely keeps some *outlines* of the objects and usually loses fine-grained *textures*. In comparison, it is identified by ColorCNNs that both *shapes* (e.g., boat hull and vehicle outline in Fig. 10, airplane wings and dog head in Fig. 11) and *textures* (e.g., cat tabby and face, vehicle wheels and windshield, and bird beak and breast in Fig. 10 and Fig. 11) are critical for single-label classifiers to correctly identify the image. Such findings also well align with some previous works [54], [57] that both shapes and textures help neural network recognition.

Visualization results in large color spaces. However, as the color space size increases, pixels naturally support more structures (e.g., Fig. 11 (d), MedianCut results under 5-bit color space). Prioritizing key structures, visual fidelity of the ColorCNN quantized images (e.g., Fig. 11 (e)-(g)) does not increase at a similar pace as the traditional methods when color space sizes increase. On the other hand, as shown in Fig. 11 (j), ColorCNN+ quantization results under 5-bit color spaces are significantly more visually accurate compared to ColorCNN outputs, and comparable to the traditional methods. With that said, ColorCNN+ results are still less visually accurate (e.g., dog before the blue sky) as it preserves not only visual fidelity under large color spaces but also key structures under small color spaces.

4.2.2 Recognition Accuracy

Recognition accuracy in small color spaces. As shown in Fig. 12, for 1-bit and 2-bit color spaces, both ColorCNN and ColorCNN+ achieve significantly better performance on different datasets using different networks. These results verify that the structures identified and preserved by ColorCNNs are indeed critical for neural network recognition.

Recognition accuracy in large color spaces. Under 5-bit and 6-bit color spaces, ColorCNN cannot compete with traditional color quantization methods in terms of network recognition accuracy. In fact, richer colors naturally support more structures, making the less visually accurate

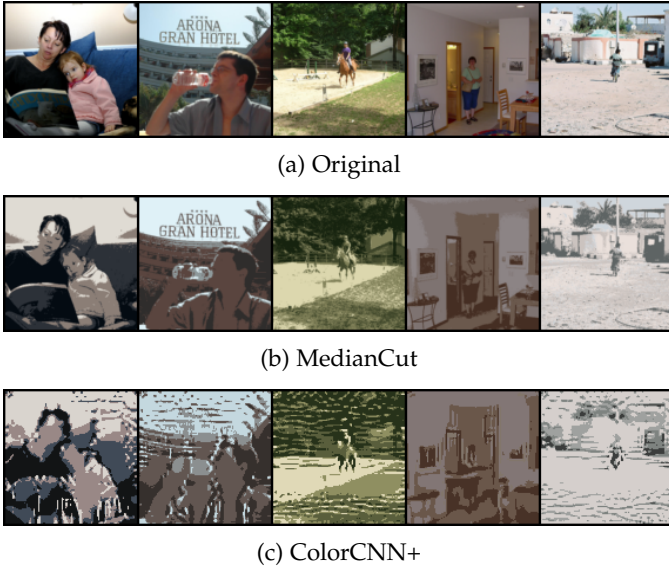


Fig. 13: Color quantization results (2-bit) for multi-label classifier on Pascal VOC 2012 dataset.

TABLE 2: Multi-label classification accuracy (%) and mean average precision (mAP, %) on Pascal VOC 2012 dataset using ResNet18 multi-label classifier. On original images, the classifier achieves 30.5% accuracy and 55.3% mAP.

color space size (bits)		1	2	3	4	5	6
accuracy	MedianCut	9.6	18.0	23.4	26.6	28.3	29.1
	ColorCNN+	13.1	21.5	25.8	29.1	30.2	31.0
mAP	MedianCut	24.5	39.6	46.7	50.4	52.4	53.5
	ColorCNN+	33.4	41.1	44.9	49.7	51.9	53.2

results from ColorCNN (Fig. 11 (g)) less competitive. In contrast, higher visual fidelity from ColorCNN+ results enables much easier recognition under *large* color spaces. ColorCNN+ not only greatly outperforms vanilla ColorCNN, but also is competitive to the traditional color quantization methods like MedianCut, which we used as the ground-truth in our imitation cluster learning.

4.3 Discussion

4.3.1 Images with Multiple Salient Objects

Quantitatively speaking (see Table 2), ColorCNN+ shows its effectiveness in terms of multi-label classification accuracy (predicting all labels correctly), outperforming MedianCut on most settings. In terms of mean average precision (mAP), which is evaluated based on precision and recall at different sigmoid thresholds for each individual label, ColorCNN+ remains competitive, but falls slightly behind MedianCut at higher bit rates. In fact, as the multi-label classification loss pushes the ColorCNN+ output towards being more confident, evaluation at multiple thresholds (e.g., mAP) can be more demanding than evaluation at the default sigmoid threshold of 0.5 (e.g., multi-label classification accuracy).

Qualitatively speaking (see Fig. 13), ColorCNN+ shows that for more complicated scenes (multiple objects-of-interest of different shapes and sizes, e.g., person, horse, motorbike, desk, bottle), the multi-label classifier prioritizes

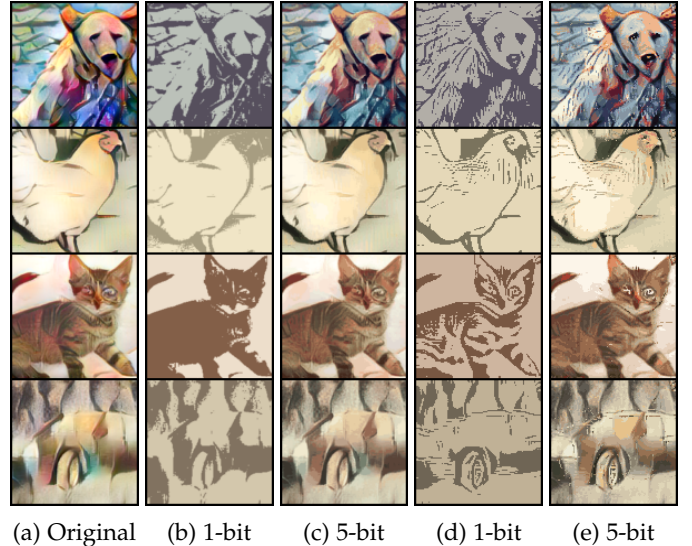


Fig. 14: Color quantization results on stylized images. (a) is the original image. (b) and (c) are MedianCut results. (d) and (e) are ColorCNN+ results.

TABLE 3: Classification accuracy (%) on Style14 dataset using ResNet18 classifier. On original (stylized but not color quantized) images, the classifier achieves 72.0% accuracy.

color space size (bits)	1	2	3	4	5	6
MedianCut	36.4	59.3	66.9	67.3	68.1	70.6
ColorCNN+	66.4	69.7	71.0	70.9	71.8	71.7

the coarse-grained overall shapes. In comparison, for single-label classification, as the classifiers only need to identify one object-of-interest, they additionally use fine-grained details like textures to further boost their performance. The multi-label setup is more demanding since the classifiers have to make out different objects of various shapes and sizes. Thus, the multi-label classifiers tend to focus more on the object shapes and outlines, which are more evident and easier to identify. Fine-grained details, on the other hand, are still very important. However, due to limited capacity, the classifier oftentimes cannot take full advantage of such details to further boost the performance, as reflected in the ColorCNN+ outputs.

4.3.2 Key Structures for Stylized Images

As shown in Fig. 14, compared to traditional methods like MedianCut, we find ColorCNN+ better preserves both shapes (e.g., car body, chicken head) textures (e.g., the bear fur, chicken feather, cat tabby, and car wheels). Such visual differences also translate into network recognition accuracy improvements. In Table 3, under 1-bit color spaces, ColorCNN+ outperform MedianCut by +30.0% accuracy, verifying its effectiveness.

We point out that the stylized images still have textures. Though overshadowed by the added artistic style, they are still present. Neural network classifiers learn to exploit these fine-grained structures for better performance. And thus, they are present in ColorCNN+ outputs.

TABLE 4: Variant study for ColorCNN+ with ResNet18 classifier on CIFAR100 dataset.

color palette size (bits)		1	2	3	4	5	6
MedianCut		10.2	26.1	44.0	57.8	66.2	70.8
ColorCNN		17.4	33.1	41.7	46.4	49.3	51.0
ColorCNN+	$\lambda = 0$	23.7	45.7	52.8	57.1	60.1	60.8
	$\lambda = 1$	23.0	44.5	53.1	59.1	60.6	63.0
	$\lambda = 3$ (default)	19.9	39.7	50.3	61.6	65.6	67.9
	$\lambda = 10$	15.5	30.6	44.9	59.3	66.3	70.4
	w/o bottleneck	18.9	37.3	49.8	61.4	65.1	67.8
	w/o top-K	17.9	39.0	49.8	61.0	65.7	68.1
	w/o augmentation	14.9	32.7	47.1	59.7	65.4	68.3
CE→KD		18.5	37.2	50.9	61.0	65.6	68.0

4.3.3 Structure Preferences of Different Networks

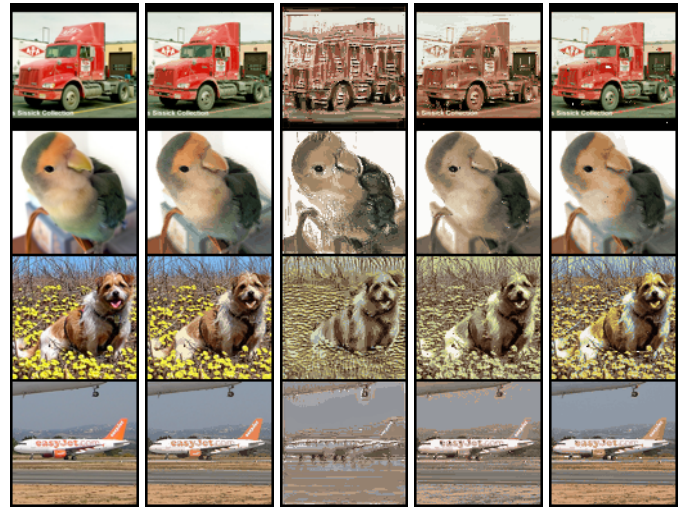
Different classifiers have different preferences in terms of coarse-grained structures (e.g., object outlines and shapes) and fine-grained structures (e.g., object details and textures). ColorCNN+ faithfully reflects such structure preferences of different networks.

First, for single-label classification (including that on both natural images and stylized images), classifiers learn to utilize *both* coarse-grained structures (e.g., outlines of boat hull and cat body in Fig. 10, airplane wing and dog head in Fig. 11, and animal bodies in Fig. 14) *and* fine-grained structures (e.g., details of cat tabby and vehicle tire in Fig. 10, monkey fur and bird beak in Fig. 11, and bear fur and chicken feather in Fig. 14). As classifiers only have to identify one object-of-interest in each image, they not only learn to identify the object shapes and outlines, but also take advantage of the informative details and textures to further improve their recognition accuracy.

Second, for the same single-label classification dataset (see Fig. 10), we find a weaker classifier like AlexNet primarily focuses on the coarse-grained shapes and cannot take full advantage of fine-grained details due to its limited capacity (e.g., no details on the bird breast and the boat deck). On the other hand, stronger classifiers like VGG16 and ResNet18 attend more on the fine-grained details in addition to the coarse-grained shapes.

Third, for multi-label classification, a task arguably more demanding than single-label classification, ColorCNN+ reflects rather different structural preferences in classifiers. As shown in Fig. 13, coarse-grained structures like outlines of human, bottle, horse, desk, and motorbike are preserved. On the other hand, fine-grained details like human eyes and mouth, or background details like signs and trees, are usually left out. A possible reason for this is that the multi-label classification might turn out to be too demanding for the classifier. Although those details might help the recognition task, their limited capacity can prevent classifiers from taking full advantage of those details.

Combing the aforementioned observations, we have the following findings. First, both coarse-grained and fine-grained structures help network recognition (coherent with previous findings in [54], [57]). Second, coarse-grained shapes and outlines are usually easier to learn and thus more commonly seen. Fine-grained details and textures, although helpful, can be difficult to learn for weak classifiers and demanding tasks due to limited network capacity.



(a) Original (b) MCut (c) $\lambda = 0$ (d) $\lambda = 3$ (e) $\lambda = 10$

Fig. 15: 6-bit color quantization results. “MCut” in (a) is short for MedianCut. (b)-(e) are ColorCNN+ variants. $\lambda = 3$ is the default hyper-parameter setting in ColorCNN+.

4.3.4 Key Structures: for Networks and for Humans

In **small** color spaces, visualizations of ColorCNN+ results verify that both coarse-grained and fine-grained structures help neural network recognition. On the other hand, for human viewing, dithering generates a noise pattern to add details, but harms neural network recognition accuracy (Fig. 12). Comparing ColorCNNs (for network) and dithering (for human), we find the fine-grained structures like textures and details help both, but are rather distinct for their recognition.

4.3.5 Key Structures vs. Accurate Colors

In **large** color spaces, increasing the weight λ for imitating traditional clustering-based methods effectively increases ColorCNN+ visual fidelity (Fig. 15) and recognition accuracy (Table 4). This raises a question: *is preserving visual fidelity (accurate colors) always beneficial to network recognition?*

The answer is No. As shown in Fig. 15 and Table 4, increasing the weight λ for the relationship preserving loss advocates visual fidelity, but does not always lead to higher recognition accuracy. In fact, for smaller color spaces (1 or 2 bits), increasing λ actually decreases recognition accuracy. This aligns with our previous findings that ColorCNNs perform better than traditional color quantization method (e.g., MedianCut), even though traditional method results are still arguably more visually similar in terms of the pixel differences. It is also suggested that there exists a trade-off between these two concepts in small color spaces, and prioritizing the informative structures actually leads to higher neural network recognition accuracy. It is until larger color spaces (5 or 6 bits) that promoting the visual fidelity consistently improves the recognition accuracy, as the large color spaces naturally supports the structures. This justifies our choice of an intermediate weight for clustering loss $\lambda = 3$ in all experiments.

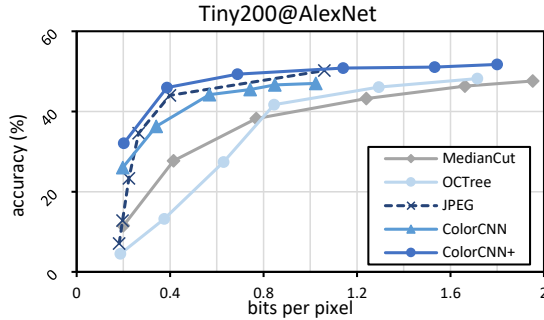


Fig. 16: Classification accuracy under different bitrate. Solid lines refer to color quantized image encoded via PNG. The dotted line refers to JPEG encoding as a reference. For color quantization methods, bitrate from low to high are quantized images under color space size from 1-bit to 6-bit.

4.3.6 Low-bit Recognition Accuracy of Different Classifiers

Color quantized images can achieve higher recognition accuracy with a weaker classifier. We compare the recognition accuracy with different classification networks in different rows of Fig. 12. It is found that a stronger classifier can have lower accuracy in an extremely small color space. For instance, on Tiny200 dataset, in 1-bit color space, AlexNet, VGG16, and ResNet18 recognize the MedianCut results with 11.3%, 10.2%, and 6.2% accuracy, respectively. Stronger classifiers can apply stronger transformation to the image data, extracting more expressive features, thus having higher accuracy for full-color images. However, when the color space is limited, stronger transformation can lead to larger drifts in feature space, leading to poor generalizability.

4.4 Applications

ColorCNNs as image compression. In Fig. 16, as the color space size grows from 1-bit to 6-bit, the quantized images take a higher bitrate when encoded with PNG, and are better recognized. When compared to traditional color quantization methods, ColorCNN can reach higher test accuracy under a lower bitrate. Moreover, under 0.2 bits per pixel, 1-bit ColorCNN quantization can even outperform JPEG compression by +13.2%, which has arbitrary number of colors. ColorCNN+, on the other hand, constantly outperforms not only all other color quantization methods, but also JPEG image compression in terms of accuracy-to-compression ratio. This clearly demonstrates the effectiveness of ColorCNN+. In fact, under 0.2 bits per pixel, 1-bit ColorCNN+ results outperform JPEG compression by +19.3%. Moreover, ColorCNN+ supports multiple color space size configurations with a single model. Compared to vanilla ColorCNN, ColorCNN+ takes more bits to encode under larger color spaces, as the color spaces are more effectively utilized in ColorCNN+. Compared to other deep image compression methods that require a neural network decoder, the ColorCNNs with PNG encoding have minimal decoding computation requirement.

ColorCNNs as adversarial defense. Adversarial attacks add small perturbations to fool neural networks [58]. Lossy image compression like JPEG can effectively defend against such attacks [59]. ColorCNNs, as lossy image compression

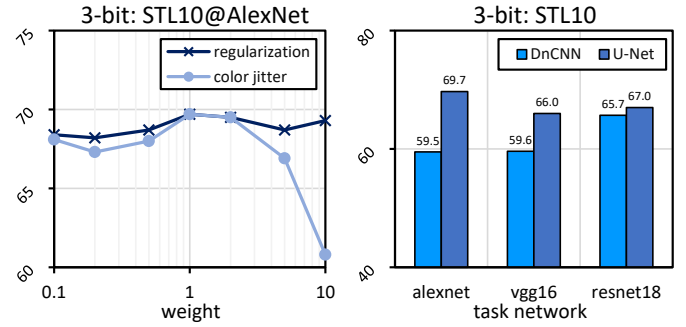


Fig. 17: ColorCNN performance (%) comparison with different weights and different auto-encoder backbone.

TABLE 5: ColorCNN and its variants under 3-bit color space, STL10 dataset, AlexNet classifier.

	Accuracy (%)	#color/image	#bit/pixel
ColorCNN	69.7	8.0	0.425
w/o regularization	67.5	5.1	0.323
w/o color jitter	67.8	8.0	0.390

methods, can also help defend against adversarial attacks. See appendix for experiments on adversarial defense.

4.5 Ablation and Variant Study

4.5.1 ColorCNN

Influence of color jitter. As shown in Table 5, without color jitter, the train-time quantization can be too easy for the pre-trained classifier. This can lead to overfitting, which further hurts accuracy by -1.9%. Too small or too large a color jitter can also result in a huge accuracy drop (Fig. 17). This is because setting the weight too small or too large leads to either limited influence, or overshadowing anything else.

Influence of color maximum regularization. We find that removing the color maximum regularization term causes an accuracy drop in Table 5. In fact, without the regularization, fewer colors are chosen during test-time. This is because the softmax color filling during training can introduce more colors in the image, as shown in Fig. 7. When the regularization weight is too small or too high, ColorCNN performance decreases (Fig. 17).

Influence of feature extractor backbone. As shown in Fig. 17, when the auto-encoder backbone is replaced with DnCNN [60], the ColorCNN performance degrades under all classification networks. Unlike U-Net, DnCNN does not have bypasses to maintain the fine-grained structures. As a result, its quantization results might have structure misalignment, which hurts classification accuracy.

4.5.2 ColorCNN+

Influence of the bottleneck layer. As in Table 4, the removal of the bottleneck layer results in accuracy drops in small and medium-sized color spaces (-2.0%, -0.7%, -0.5%, and -0.6% for 1-4 bit color spaces), and slight improvements in large color spaces (+0.1% and +0.2% for 5 and 6-bit color spaces). The performance drops confirms that redundant and irrelevant information can lead to overfitting under small color spaces. On the other hand, the performance increases under

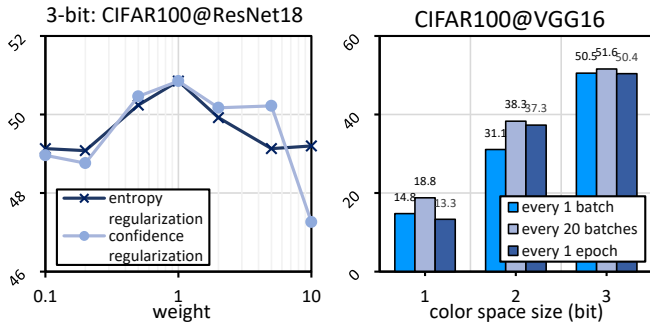


Fig. 18: **Left:** ColorCNN+ performance comparison with different weights for R_{info} and R_{conf} . **Right:** Training ColorCNN+ using different frequency for task selection update.

large color spaces align with our understanding that more difficult decisions benefit from more information. Overall, the performance differences under small and medium-sized color spaces largely outweigh those in large color spaces, proving the effectiveness of this low-dimension bottleneck.

Influence of K non-zero terms. We find that removing the top- K non-zero mask in the color probability map $m_C(x)$ leads to consistent performance drops in Table 4. This verifies its effectiveness in combating overfitting, as it directly forces the train-time forward pass approximation to behave more like the test-time counterpart.

Influence of the pixel-wise relationship preserving loss. In Fig. 15 and Table 4, we find that the removal of the imitation learning loss results in undesirable visual quality (limited usage of the color spaces) and inferior recognition accuracy under large color spaces, verifying its effectiveness. With that said, the inclusion of the pixel-wise relationship preserving loss also results in accuracy drops for small color spaces. In this regard, we refer readers to Section 4.3, where we discuss the trade-off between accurate colors and key structure, especially in small color spaces.

Information entropy and confidence regularization terms. In Fig. 18 left, we find that too small or too large values of the two regularization terms lead to worse performance. These results probably due to too little regularizations are not strong enough, to combat the limited color choices (information entropy regularization) or overfitting (confidence regularization). On the other hand, while too much regularizations can overshadow other loss components like the cross-entropy loss or the pixel-wise relationship preserving loss. We settle at setting $\alpha = 1$ and $\beta = 1$ for all our experiments.

Pace of task selection change. As ColorCNN+ natively supports multiple color space sizes in one model, during training, how to choose the color space size becomes an important question. This is referred as task selection (see Section 3.6.3). As shown in Fig. 18, we find that neither too fast (*i.e.*, every batch) nor slow (*i.e.*, every epoch) the pace in task change gives better results than the proposed intermediate pace for changing the task (color space size) in training. This aligns with the previous study [45] on task selection for video understanding.

Augmentation on the quantized images. As shown in Table 4, the absence of post-time augmentations on quan-

tized images results in performance drops except for 6-bit color spaces. Especially, for small color spaces (1 and 2-bit), the resulting variant is even outperformed by the vanilla ColorCNN. This verifies that the augmentation is vital for learning the informative structures in ColorCNN+, and that augmentations on the quantized images effectively prevent overfitting in ColorCNN+ training.

Knowledge distillation instead of cross-entropy loss.

In the absence of ground-truth labels for the images, we can replace the cross-entropy loss in Eq. 13 with knowledge distillation loss [61], which uses the classifier output on the original image as soft targets. As shown in Table 4, using knowledge distillation loss instead of cross-entropy loss achieves similar performance.

5 CONCLUSION

In this paper, we investigate the scientific problem of keeping informative structures with limited colors. Specifically, it is found that using the task network and a specifically designed architecture, ColorCNN, we can preserve the informative structures to enable neural network recognition in small color spaces. By imitating traditional clustering-based color quantization methods, ColorCNN+, an updated architecture that supports multiple color space size configurations can output visually accurate results under large color spaces. Extensive experiments showcase the identified informative structures and enable study on when and how the informative structures and accurate colors help neural network recognition.

REFERENCES

- [1] P. Heckbert, *Color image quantization for frame buffer display*. ACM, 1982, vol. 16, no. 3.
- [2] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.
- [3] M. T. Orchard and C. A. Bouman, "Color quantization of images," *IEEE transactions on signal processing*, vol. 39, no. 12, pp. 2677–2690, 1991.
- [4] R. Floyd and L. Steinberg, "An adaptive technique for spatial grayscale," in *Proceedings of the Society of Information Display*, vol. 17, 1976, pp. 78–84.
- [5] M. Gervautz and W. Purgathofer, "A simple method for color quantization: Octree quantization," in *New trends in computer graphics*. Springer, 1988, pp. 219–231.
- [6] Y. Hou, L. Zheng, and S. Gould, "Learning to structure an image with few colors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 116–10 125.
- [7] Y. Deng, C. Kenney, M. S. Moore, and B. Manjunath, "Peer group filtering and perceptual color image quantization," in *ISCAS'99. Proceedings of the 1999 IEEE International Symposium on Circuits and Systems VLSI (Cat. No. 99CH36349)*, vol. 4. IEEE, 1999, pp. 21–24.
- [8] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.
- [9] Y. Deng and B. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE transactions on pattern analysis and machine intelligence*, vol. 23, no. 8, pp. 800–810, 2001.
- [10] X. Wu, "Color quantization by dynamic programming and principal analysis," *ACM Transactions on Graphics (TOG)*, vol. 11, no. 4, pp. 348–372, 1992.
- [11] T. Boutell and T. Lane, "Png (portable network graphics) specification version 1.0," *Network Working Group*, pp. 1–102, 1997.
- [12] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

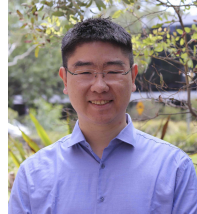
- [13] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *IEEE Signal processing magazine*, vol. 18, no. 5, pp. 36–58, 2001.
- [14] C. Poynton, *Digital video and HD: Algorithms and Interfaces*. Elsevier, 2012.
- [15] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv preprint arXiv:1601.06759*, 2016.
- [16] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. Jin Hwang, J. Shor, and G. Toderici, "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4385–4393.
- [17] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5306–5314.
- [18] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, "Practical full resolution learned lossless image compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 629–10 638.
- [19] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning convolutional networks for content-weighted image compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3214–3223.
- [20] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves et al., "Conditional image generation with pixelcnn decoders," in *Advances in neural information processing systems*, 2016, pp. 4790–4798.
- [21] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool, "Generative adversarial networks for extreme learned image compression," *arXiv preprint arXiv:1804.02958*, 2018.
- [22] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.
- [23] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," *arXiv preprint arXiv:1802.01436*, 2018.
- [24] Z. Liu, X. Xu, T. Liu, Q. Liu, Y. Wang, Y. Shi, W. Wen, M. Huang, H. Yuan, and J. Zhuang, "Machine vision guided 3d medical image compression for efficient transmission and accurate segmentation in the clouds," *arXiv preprint arXiv:1904.08487*, 2019.
- [25] X. Wei, I. A. Barsan, S. Wang, J. Martinez, and R. Urtasun, "Learning to localize through compressed binary maps," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 316–10 324.
- [26] F. Camposeco, A. Cohen, M. Pollefeys, and T. Sattler, "Hybrid scene compression for visual localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7653–7662.
- [27] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.
- [28] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *international conference on machine learning*. PMLR, 2017, pp. 3861–3870.
- [29] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*, 2016, pp. 478–487.
- [30] F. Li, H. Qiao, and B. Zhang, "Discriminatively boosted image clustering with fully convolutional auto-encoders," *Pattern Recognition*, vol. 83, pp. 161–173, 2018.
- [31] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," *arXiv preprint arXiv:1801.07648*, 2018.
- [32] C.-C. Hsu and C.-W. Lin, "Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data," *IEEE Transactions on Multimedia*, vol. 20, no. 2, pp. 421–429, 2017.
- [33] Z. Wang, S. Chang, J. Zhou, M. Wang, and T. S. Huang, "Learning a task-specific deep architecture for clustering," in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 369–377.
- [34] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5147–5156.
- [35] D. Chen, J. Lv, and Y. Zhang, "Unsupervised multi-manifold clustering by learning deep representation," in *Workshops at the thirty-first AAAI conference on artificial intelligence*, 2017.
- [36] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang, "Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5736–5745.
- [37] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, "Learning discrete representations via information maximizing self-augmented training," in *International conference on machine learning*. PMLR, 2017, pp. 1558–1567.
- [38] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
- [39] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *2014 22nd International conference on pattern recognition*. IEEE, 2014, pp. 1532–1537.
- [40] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1365–1374.
- [41] Z. Li, R. Jiang, and P. Aarabi, "Semantic relation preserving knowledge distillation for image-to-image translation," in *European conference on computer vision*. Springer, 2020.
- [42] Y. Hou and L. Zheng, "Visualizing adapted knowledge in domain transfer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 824–13 833.
- [43] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.
- [44] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 13 001–13 008.
- [45] C.-Y. Wu, R. Girshick, K. He, C. Feichtenhofer, and P. Krahenbuhl, "A multigrid method for efficiently training video models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 153–162.
- [46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [47] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [49] A. Krizhevsky, G. Hinton et al., "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.
- [50] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.
- [51] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, 2015.
- [52] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [53] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.
- [54] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *arXiv preprint arXiv:1811.12231*, 2018.
- [55] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. International Society for Optics and Photonics, 2019, p. 1100612.
- [56] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.
- [57] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [58] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural

networks,” in *International Conference on Learning Representations*, 2014. [Online]. Available: <http://arxiv.org/abs/1312.6199>

- [59] N. Das, M. Shanbhogue, S.-T. Chen, F. Hohman, S. Li, L. Chen, M. E. Kounavis, and D. H. Chau, “Shield: Fast, practical defense and vaccination for deep learning using jpeg compression,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 196–204.
- [60] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising,” *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.
- [61] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.



Yunzhong Hou received his bachelor degree in electronic engineering from Tsinghua University in 2018. He is now working towards a PhD degree at Australian National University under the supervision of Dr. Liang Zheng and Prof. Stephen Gould. His research interests lie in computer vision and deep learning.



identification.

Liang Zheng is a Lecturer and a Computer Science Futures Fellow in the Research School of Computer Science, Australian National University. He received the PhD degree in Electronic Engineering from Tsinghua University, China, in 2015, and the B.E. degree in Life Science from Tsinghua University, China, in 2010. He was a postdoc researcher in the Center for Artificial Intelligence, University of Technology Sydney, Australia. His research interests include image retrieval, classification, and person re-



sion, machine learning, probabilistic graphical models, and optimization.

Stephen Gould received the BSc degree in mathematics and computer science and BE degree in electrical engineering from the University of Sydney, in 1994 and 1996, respectively, the MS degree in electrical engineering from Stanford University, in 1998, and the PhD degree from Stanford University, in 2010. He is a professor with the Research School of Computer Science, College of Engineering and Computer Science, Australian National University. His research interests are in computer and robotic vi-