Exploring Image Representation with Decoupled Classical Visual Descriptors (Supplementary Materials)

Chenyuan Qu[1,3]
cxq134@student.bham.ac.uk
Hao Chen[1,2]
hc666@cam.ac.uk
Jianbo Jiao[1]
j.jiao@bham.ac.uk

[1] The MIx Group
University of Birmingham, UK
[2] University of Cambridge
Cambridge, UK
[3] Allsee Technologies Ltd
Birmingham, UK

This supplementary document provides additional details and results to support the main paper. In Section 1, we discuss the artistic perspective of images, linking our proposed descriptors to artistic elements. Section 2 describes the experimental settings, including training configurations, optimisation strategies for multi-task learning, and choice of colour segmentation clusters. Section 3 introduces and derives the information-theoretic analysis of different model inputs. In Section 4, we present an ablation study analysing the contributions of different loss functions. Section 5 showcases additional representation experiments, including qualitative analysis on colour editing task and representation experiment on brightness editing task. Section 6 discusses social impact, section 7 shows the experiments computing resources. Finally, section 8 displays more visualisation results on image restoration task (in subsection 8.1) and image colour editing task (in subsection 8.2).

1 Artistic Perspective on Images

The decoupling of image representation in computational visual perception is an important problem. Human visual perception, being inherently abstract, also relies on the ability to identify and explain fundamental components within a visual scene. To introduce a method for describing human visual scenes, *Esaak* [4] proposes a method to deconstruct images into seven basic artistic elements: line, shape, form, space, texture, value, and colour (see Figure 1). They function as the building blocks for composing an image and offer a thorough framework for understanding and describing the image at hand.

This paper aims to model the aforementioned abstract nature of human perception within the framework of the proposed learning paradigm. Previous studies have made significant progress in utilising classical visual descriptors to accomplish vision tasks using deep learning methods, such as leveraging shape and texture for classification tasks [8], as well as incorporating shape, texture, and colour for recognition purposes [5]. In this paper, we aim to explore how a more generalised visual perception can be learned from classical visual descriptors in a learning-based framework, focusing on three fundamental descriptors: edge,

grey-scale intensity, and colour. We highlight that colour directly represents itself, and grey-scale intensity reflects form and value. Meanwhile, edges offer more generalised geometric information and encapsulate aspects of line, space, and texture. Collectively, these three descriptors encompass the seven fundamental elements outlined in Esaak's theory, forming a comprehensive foundation for visual analysis.

Notes: Although the proposed descriptors, such as the Sobel operator, may be considered outdated and less effective, this work aims to establish a foundational approach for leveraging *'basic'* descriptors to enable the learning of more advanced representations, as confirmed by the experiments in the paper.

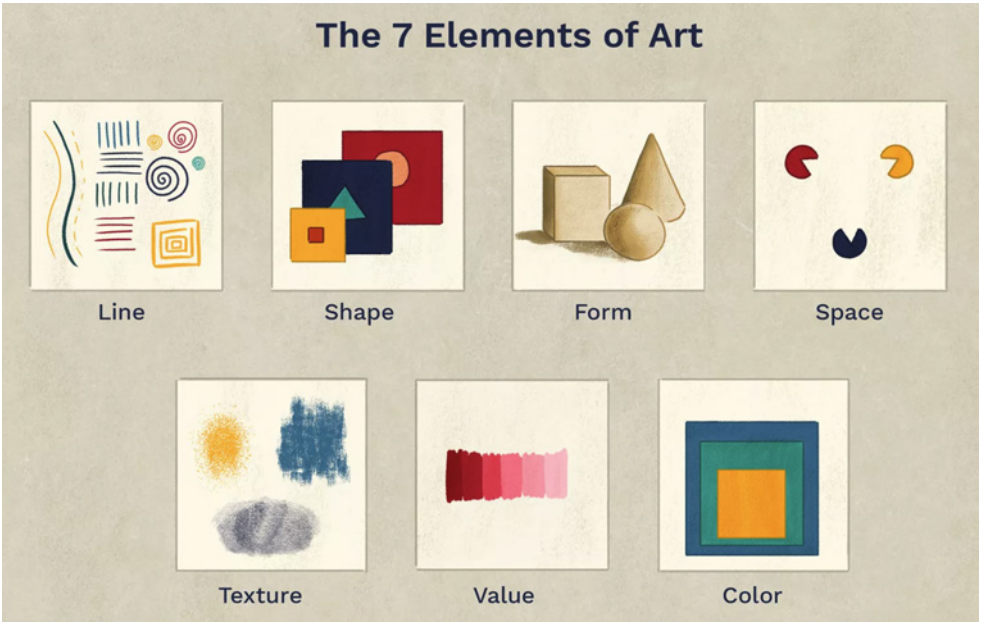


Figure 1: Depiction of the 7 elements of art (figure from [4]).

2 Experiment Settings

We have specified disclosure hyperparameters for each experimental configuration as detailed in Table 1. Drawing from the ViT training setup as a basis, we conducted a preliminary parameter experiment to establish the optimal hyperparameters for each individual experiment.

2.1 Multi-Task Learning

Training with multiple tasks presents a unique challenge: how to balance and coordinate them effectively. Introducing all losses from the outset proves counterproductive, as the early training stages often produce disorganised reconstructions, rendering the descriptor consistency loss irrelevant. To address this, we adopt an *optimisation-scheduling* approach,

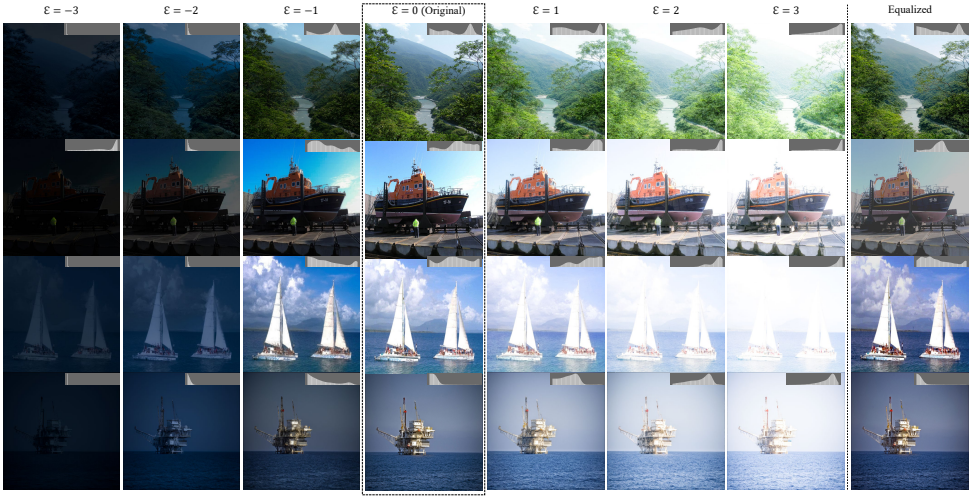


Figure 2: Comparison of brightness editing results using modified grey-level histograms as guidance input with Stable Diffusion 1.5 (v_prediction) with our method. Each column represents a different exposure compensation, ε, with values ranging from -3 to 3, followed by an equalized version. The original images are displayed in the central column. Each image's corresponding grey-level histogram is shown in the top right corner of each image.

delaying the introduction of the descriptor consistency loss until after *50* epochs, when the reconstructed images become more coherent and meaningful.

Another challenge lies in the varying scales of the loss functions, which can impede their effective integration. To address this, we normalise each loss by its maximum observed value during training, ensuring they operate on a common scale. Additionally, we employ an automatic weight adjustment strategy that dynamically allocates weights to each normalised loss based on its relative importance and contribution during training. The weight w_i for each loss at time t is calculated as:

$$w_i(t) = \frac{\exp\left(\frac{\Delta L_i(t)}{T}\right)}{\sum_j \exp\left(\frac{\Delta L_j(t)}{T}\right)}, \tag{1}$$

where the term $\Delta L_i(t)$ quantifies the change at iteration t relative to the previous step, with the temperature parameter $T = 2.0$ controlling the sensitivity of weight adjustments.

2.2 Implementation of Differentiable Descriptors

2.2.1 Sobel Operator

We compute image gradients with a differentiable Sobel operator in three steps:

1. **Kernel definition.** Two fixed 3×3 kernels capture horizontal and vertical intensity changes:

$$\mathbf{K}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad \mathbf{K}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

Table 1: Implementation hyper-parameter details.

Config	Pre-training	Fine-tuning	Linear Probing
optimizer	AdamW	AdamW	AdamW
learning rate	1.5e-4	1.0e-3	1.0e-1
weight decay	1e-5	1e-5	0
β_1	0.95	0.95	0.9
β_2	0.999	0.999	0.999
ε	1e-8	1e-8	1e-8
batch size	128	128	256
learning rate scheduler	cosine decay	cosine decay	cosine decay
warmup steps	1000	1000	500
image size	224	224	224
data normalisation	✓	✓	✗
image mean	(0.485, 0.456, 0.406)	(0.485, 0.456, 0.406)	—
standard deviation	(0.229, 0.224, 0.225)	(0.229, 0.224, 0.225)	—

Each kernel is applied channel-wise and kept constant (no learning).

2. **Convolution.** Given an input tensor $x \in \mathbb{R}^{N\times H\times W}$, we obtain per-pixel gradients by 2-D convolution with unit stride and padding 1:

$$g_x = \text{conv2d}(x, \mathbf{K}_x), \qquad g_y = \text{conv2d}(x, \mathbf{K}_y).$$

3. **Gradient magnitude.** The final edge strength combines the two orthogonal responses:

$$e = \sqrt{g_x^2 + g_y^2} \in \mathbb{R}^{N\times H\times W}.$$

2.2.2 Soft K-means

We assign features to clusters with a differentiable, temperature-controlled variant of k-means:

1. **Learnable centroids.** We maintain M cluster centres $\mathbf{C} = \{\mathbf{c}_m\}_{m=1}^{M} \in \mathbb{R}^{M\times D}$ as trainable parameters, where D is the feature dimension.

2. **Pairwise distances.** For a mini-batch of feature vectors $\mathbf{X} = \{\mathbf{f}_i\}_{i=1}^{B} \in \mathbb{R}^{B\times D}$ we compute the Euclidean distance to every centroid:

$$d_{im} = \left\|\mathbf{f}_i - \mathbf{c}_m\right\|_2, \qquad \mathbf{D} \in \mathbb{R}^{B\times M}.$$

3. **Soft assignment with temperature.** Distances are converted to assignment probabilities through a softmax scaled by an inverse temperature τ:

$$a_{im} = \frac{\exp(-d_{im}/\tau)}{\sum_{m'=1}^{M}\exp(-d_{im'}/\tau)}, \qquad \mathbf{A} \in \mathbb{R}^{B\times M}.$$

A lower τ yields crisper assignments, while $\tau \to 0$ recovers hard k-means. In our case, we set $\tau = 1e-3$.

Table 2: Comparative performance with different number of decoder blocks in Linear Probing and Finetuning

Block	1	2	4	8	12	24
Linear Probing	60.2	68.8	72.5	**74.0**	73.6	73.4
Fine-tuning	82.9	83.4	83.4	**83.5**	83.4	83.4

2.2.3 Grey-level Histogram

We implement the differentiable grey-level histogram through the following steps:

1. **Binning:** We define K bins uniformly spaced within the grey-level interval $[0,1]$:

$$b_k = \frac{k-1}{K-1}, \quad k = 1,2,\ldots,K.$$

2. **Gaussian Kernel Smoothing:** For each pixel value x_{np} (from the input image tensor $x \in \mathbb{R}^{N\times H\times W}$), we compute its distance to each bin centre b_k, and apply a Gaussian kernel with standard deviation σ:

$$G(x_{np}, b_k; \sigma) = \exp\left(-\frac{(x_{np}-b_k)^2}{2\sigma^2}\right).$$

3. **Aggregation:** Finally, we sum the kernel responses across the spatial dimension ($H \times W$) to form the smooth histogram h_{nk}:

$$h_{nk} = \sum_{p=1}^{HW} G(x_{np}, b_k; \sigma).$$

2.3 Choice of Colour Segmentation Clusters

In Section 4.2 of the main paper, we have elaborated on the rationale behind our choice of the number of colour segmentation clusters. This hyperparameter was consistently applied across all tasks, and our results on various tasks ,as well as the Places and ADE20K datasets, demonstrate the generality of our selection. However, we need to note that this value can be treated as a hyperparameter and adjusted for different tasks and datasets. Such adjustments, while potentially valuable, are beyond the scope of this study.

2.4 The Power of Image Decoder

We also investigated the impact of the decoder's capacity on performance. We compared the decoders with different numbers of blocks in Table 2. It shows 8-block decoder performs the best. Although a stronger decoder may lead to better pixel for the pre-train reconstruction task, the performance on the latent representation is more important for the downstream task. ViTMAE [6] also suggested that a lightweight decoder benefits the encoder learning more in the pre-training task.

Table 3: Comparative performance with different latent vector length in Linear Probing and Finetuning

Length	128	256	512	768	1024
Linear Probing	61.5	68.9	73.5	74.0	**74.6**
Fine-tuning	75.1	80.1	83.3	83.5	**83.6**

2.5 Further Research on Latent Vectors

Although our goal is not to achieve the best performance, and thus we simply followed the ViT-base [3] design for the latent vector length, we still conducted a comparison of different latent vector lengths, with the results shown in Table 3, and we found that the larger the latent vector length, the better the performance, up to a length of 1024.

3 Shannon Entropy and Representation Learning

Because the self-supervised representation learning approaches commonly rely on the principle of missing information prediction, the input complexity is necessary to compare different methods. Based on information Bottleneck method [13] - the model should capture the most representative essence of the input in order to fulfil its task when it receives only partial information, increasing the amount of missing information generally enhances the difficulty of the representation learning task and encourages the model to learn more meaningful representations. [1] We can justify it via **Shannon Entropy** [12]. Let's revise the Shannon Entropy definition, for a discrete variable X,

$$H(X) = -\mathbb{E}_x \left[\log P(x)\right]. \tag{2}$$

In the context of missing information, the uncertainty of the missing content given the observed input is measured by the conditional entropy:

$$H(M \mid O) = -\mathbb{E}_{P(M,O)} \left[\log P(M \mid O)\right], \tag{3}$$

where M stands for missing information, O is the observed information. When we try to minimise the prediction loss during the pre-training task, we are essentially approaching the optimal $P(M|O)$.

Therefore, when M increases, while O decreases, the entropy $H(M \mid O)$ generally increases, making the prediction task more difficult. To reduce this uncertainty, the model is forced to extract higher-level semantic and structural features from the observed content O, thereby yielding more abstract and meaningful representations.

To quantitatively compare the input complexity and theoretical efficiency of different pre-training methods, we adopt the concept of *Maximum Shannon Entropy*. The Maximum Shannon Entropy represents the highest possible information content (in bits) an input representation can theoretically encode, assuming all input symbols are independent and uniformly distributed. This provides a method-agnostic metric to assess the theoretical upper bound of information provided by the input data.

3.1 General Formula

Given an input consisting of discrete symbols, each represented by a fixed number of bits, the Maximum Shannon Entropy (in bits) is computed as:

$$H_{\max} = \sum_i N_i \cdot b_i \tag{4}$$

where N_i is the total number of symbols of the i^{th} type, and b_i is the number of bits used to represent each symbol.

When the input consists primarily of RGB pixels, each pixel has three colour channels (RGB), and each channel typically has 8 bits:

$$H_{\text{RGB}}^{\max} = N \times C \times b \tag{5}$$

where:

- N is the number of pixels (e.g., 224×224),
- $C = 3$ for RGB channels,
- $b = 8$ bits per channel.

3.2 Instantiation for Each Method

We instantiate the above general formula (Eq. 4) for specific methods:

(1) Raw RGB Images For an RGB image of resolution 224×224:

$$H_{\text{RGB}}^{\max} = 224 \times 224 \times 3 \times 8 = 1,204,224 \text{ bits} \approx 1.204 \text{ Mbit} \tag{6}$$

(2) Split-Brain Autoencoder Split-Brain Autoencoder [15] splits input into two branches: one with the L channel (greyscale) and the other with ab channels (colour). The average entropy across branches is thus:

$$H_{\text{SBA}}^{\max} = \frac{1}{2}(224 \times 224 \times 1 \times 8 + 224 \times 224 \times 2 \times 8) = \frac{1}{2} H_{\text{RGB}}^{\max} \approx 0.602 \text{ Mbit} \tag{7}$$

(3) PeCo (Perceptual Codebook) [2] PeCo applies a masking strategy similar to BEiT, masking 40% of pixels and thus retaining 60% of pixels:

$$H_{\text{PeCo}}^{\max} = 0.6 \times H_{\text{RGB}}^{\max} \approx 0.723 \text{ Mbit} \tag{8}$$

(4) MAE (Masked Autoencoder) [6] MAE masks 75% of pixels, keeping only 25%:

$$H_{\text{MAE}}^{\max} = 0.25 \times H_{\text{RGB}}^{\max} \approx 0.301 \text{ Mbit} \tag{9}$$

(5) VisualSplit (Ours) Our VisualSplit uses compact descriptors rather than raw RGB:

- Binary edge map: 1 bit/pixel
- 8-cluster colour map: $\log_2(8) = 3$ bits/pixel
- Grey-level histogram (100 bins, 8 bits/bin): 100×8 bits

Table 4: Maximum Shannon Entropy for each pre-training method at 224×224 resolution.

Method	Bits (Mbit)	Relative to RGB
Raw RGB	1.204	100%
Split-Brain Auto [15]	0.602	50%
PeCo [2]	0.723	60%
MAE [6]	0.301	25%
VisualSplit (Ours)	**0.202**	**17%**

Table 5: Comparative performance with and without LPIPS and MSE losses in Linear Probing and Finetuning over training steps

#steps	Linear Probing			Fine-tuning		
	ours	w/o LPIPS	w/o MSE	ours	w/o LPIPS	w/o MSE
100k	22.5	22.7 (+0.2)	17.7 (-4.8)	70.9	71.0 (+0.1)	70.0 (-0.9)
200k	42.2	42.6 (+0.4)	37.1 (-5.1)	75.7	75.6 (-0.1)	74.5 (-1.2)
400k	55.9	56.0 (+0.1)	50.3 (-5.6)	78.8	78.8 (±0.0)	78.1 (-0.7)
800k	64.7	64.6 (-0.1)	60.1 (-4.6)	81.1	80.9 (-0.2)	80.7 (-0.4)
1600k	69.6	69.2 (-0.4)	64.2 (-5.4)	82.7	82.3 (-0.4)	82.1 (-0.6)
3200k	72.6	71.9 (-0.7)	65.6 (-7.0)	83.3	83.0 (-0.3)	82.6 (-0.7)
6400k	74.0	73.0 (-1.0)	66.5 (-7.5)	83.5	83.1 (-0.4)	82.6 (-0.9)

Thus,

$$H_{\mathrm{VS}}^{\max} = 224 \times 224 \times 1 \times (1+3) + 100 \times 8 \approx 0.202 \text{ Mbit} \tag{10}$$

Because our descriptors are intrinsically sparse, we achieve this reduction without any patch masking or channel dropping, while the sparser input forces the network to model deeper underlying visual concepts, thus enabling more robust representation learning. The comparison is shown in Table 4.

4 Loss Function Ablation Study

The intuition behind the loss selection was detailed in Section 3, Methodology, Pre-training Objectives. Specifically, the combination of MSE loss and LPIPS loss is employed because our pre-training task is a reconstruction task. Furthermore, we introduced descriptor consistency loss to ensure that the model captures the input descriptor. This loss aligns the descriptors from the reconstructed image with those from the original image.

Meanwhile, we understand the importance of the individual contribution of these losses to the overall performance to justify them. Therefore, we further conducted ablation studies for them.

4.1 Reconstruction Loss

Firstly, as we mentioned on the main page, MSE loss ensures pixel-wise similarity, while the LPIPS loss measures the global similarity between the reconstructed and original images. Here, we conducted ablation studies for each. The results of those on the classification task

Figure 3: Qualitative Performance with and without LPIPS Loss. Left: Original Image, Middle: Result without LPIPS, Right: Result with LPIPS. The result without LPIPS loss has significant grid-like artefacts.

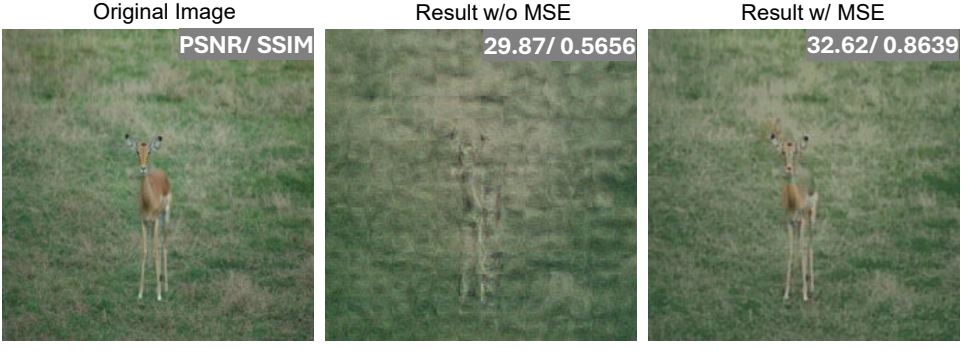


Figure 4: Qualitative Performance with and without MSE Loss. Left: Original Image, Middle: Result without MSE, Right: Result with MSE. The result without MSE loss has distorted torus-like artefacts and lose more colour information.

are shown in Table 5, and the quantitative and qualitative performance on the pre-training task are shown in Table 6 and Figure 3, respectively.

From the quantitative result, we can see that LPIPS and MSE loss both affect the performance of linear probing and fine-tuning tasks, specifically, they affect linear probing more than fine-tuning. MSE has a great impact on classification, since it may force the model to learn the details, which helps the classification task a lot. Comparatively, LPIPS loss does affect the result but not significantly, however, as shown in Figure 3, its reconstructed image has more grid-like artefacts, which is because MSE loss is not able to measure the global similarity, therefore we conduct the LPIPS loss to improve the performance to reduce the artefacts. Regarding the reconstruction result of MSE loss shown in Figure 4, the method with only the LPIPS method contains distorted torus-like artefacts [□]. The quantitative results are shown in Table 6, which shows that both reconstruction loss contributes to the pre-training task.

Table 6: Quantitative results of the pre-training image restoration task. The results are presented in PSNR/SSIM.

w/o LPIPS	w/o MSE	Full
29.71/ 0.6012	30.72/ 0.6133	**31.42/ 0.8409**

Table 7: Comparative Performance with and without descriptor loss in Linear Probing and Fine-tuning Over Training Steps. Early stopping may intervene to prevent overfitting. Descriptor loss is notated as L_d.

#steps	Linear Probing		Fine-tuning	
	ours	w/o L_d	ours	w/o L_d
100k	22.5	17.7 (-4.8)	70.9	68.9 (-2.0)
200k	42.2	34.8 (-7.4)	75.7	72.7 (-3.0)
400k	55.9	48.3 (-7.6)	78.8	76.3 (-2.5)
800k	64.7	58.5 (-6.2)	81.1	79.3 (-1.8)
1600k	69.6	63.6 (-6.0)	82.7	81.1 (-1.6)
3200k	72.6	65.8 (-6.8)	83.3	82.2 (-1.1)
6400k	74.0	66.8 (-8.0)	83.5	82.8 (-0.7)

4.2 Descriptor Loss

The descriptor loss plays a crucial role in ensuring the model effectively captures and reconstructs the input descriptors. To evaluate its contribution, we conducted ablation studies comparing the performance with and without it across both linear probing and fine-tuning tasks. The results are presented in Table 7, in which descriptor loss is notated as L_d.

For linear probing and fine-tuning, removing descriptor loss caused significant degradation in performance across all training steps. For example, at 6400k steps of linear probing, the accuracy dropped by 8.0%. The descriptor loss aligns the reconstructed image descriptors with those of the input image, ensuring that the decoupled representations remain faithful to the original visual elements.

5 More Representation Experiments for Visual Editing

5.1 Qualitative Analysis on Colour Editing Task

To evaluate the performance of our colour editing task, we conducted a survey based on three critical aspects: naturalness, accuracy, and consistency. Naturalness measures how natural the edited image looks, while accuracy shows how accurately the colour modification matches the target effect, and consistency means how well the edited image retains details from the original image.

Participants were asked to rate these attributes on a scale of 1 (poor) to 5 (excellent) for ten editing images. During the survey, the result images will be served randomly and anonymously. The result is shown in Table 8.

As shown in the table, our model achieved an average rating of 4.13 for Naturalness, 4.16 for Accuracy, and 4.25 for Consistency, significantly outperforming the other methods in all attributes.

Table 8: Survey results for different models on the image colour editing task with attributes.

Models	**Naturalness**	**Accuracy**	**Consistency**
ControlNet [14] w/ prompt	2.02	2.89	2.06
T2I Adapter [11] w/ prompt	2.72	3.10	2.13
ControlNet++ [7] w/ prompt	2.32	3.41	2.33
Ours	**4.13**	**4.16**	**4.25**

5.2 Representation Experiments on Brightness Editing Task

Continuing from the main paper Section 4.6, Representation for Visual Editing, we further explored image brightness editing through adjustments to the grey-level histogram while maintaining the same setup as before, but adjusting the grey-level histogram. Due to the page limitation, we show it here. We adjust the grey-level histogram by adjusting the exposure compensation with the following equation:

$$i' = i \times 2^{\varepsilon}, \tag{11}$$

where i is the l channel in the image. We also did histogram equalisation for the original image with the equation:

$$i' = \text{round}\left((L-1)\cdot\frac{\Sigma_{j=0}^{i} n_j}{N}\right), \tag{12}$$

where L is the number of grey-levels, because signal leakage in Stable Diffusion 1.5 training makes it only able to generate images with medium brightness [4], we used a variant with v_prediction and enabled a zero signal-to-noise ratio.

As shown in Figure 2, the edited result shows a similar effect to classic image enhancement methods while preserving the coherence of other visual elements. The generated result follows the input grey-level histogram.

6 Social Impact

Although deep representation learning is intriguing, its opacity impedes understanding of the learning process. Our approach, drawing inspiration from elements of art description, involves segmenting images into traditional descriptors using image processing methods that align with these elements. These descriptors, renowned for their human understandability, serve as a tool to spark curiosity in unravelling the black box. We strive to illuminate the inner workings of these mechanisms and advocate for transparency in the learning process.

This research paper delves into fundamental representation learning, detached from specific applications. Nonetheless, the acquired representation could potentially be harnessed for controlled image generation, serving as a potential wellspring for DeepFake attacks tailored for generating fake individuals or scenes with photorealism.

7 Experiments Computing Resources

We used four NVIDIA A100 GPUs for model pre-training, running for 40 hours. Additionally, we used a single NVIDIA A100 GPU for 12 hours to train each low-level task.

8 More Visualisation Results

8.1 More Visualisation Results on Image Restoration Task

More visualisation results on the image restoration task are shown in Figure 5 - 56.

8.2 More Visualisation Results on Image Colour Editing Task

More visualisation results on the image colour editing task are shown in Figure 57 - 61.

References

[1] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[2] Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, Nenghai Yu, and Baining Guo. Peco: Perceptual codebook for bert pre-training of vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 552–560, 2023.

[3] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] Shelley Esaak. The 7 elements of art and why knowing them is important, April 5 2023. URL https://www.thoughtco.com/what-are-the-elements-of-art-182704.

[5] Yunhao Ge, Yao Xiao, Zhi Xu, Xingrui Wang, and Laurent Itti. Contributions of shape, texture, and color in visual recognition. In *European Conference on Computer Vision*, pages 369–386. Springer, 2022.

[6] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[7] Ming Li, Taojiannan Yang, Huafeng Kuang, Jie Wu, Zhaoning Wang, Xuefeng Xiao, and Chen Chen. Controlnet++ : Improving conditional controls with efficient consistency feedback. In *European Conference on Computer Vision*, pages 129–147. Springer, 2025.

[8] Yingwei Li, Qihang Yu, Mingxing Tan, Jieru Mei, Peng Tang, Wei Shen, Alan Yuille, and Cihang Xie. Shape-texture debiased neural network training. *arXiv preprint arXiv:2010.05981*, 2, 2020.

[9] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 5404–5411, 2024.

[10] Fabian Mentzer, George D Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. *Advances in Neural Information Processing Systems*, 33:11913–11924, 2020.

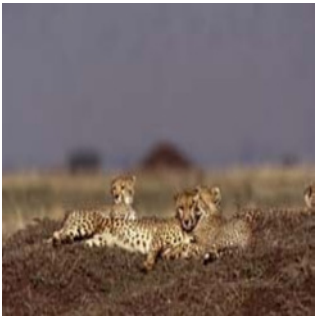
[11] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 4296–4304, 2024.

[12] Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[13] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[14] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[15] Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1058–1067, 2017.

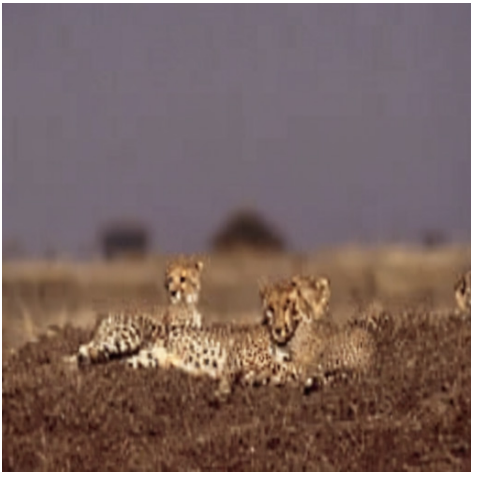


(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

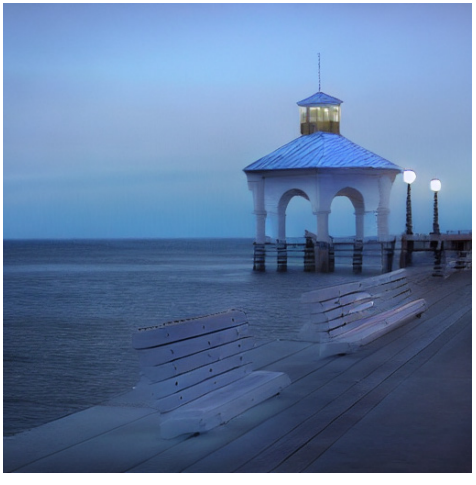
(d) ControlNet++ [7] w/ Prompt Result



(e) VisualSplit (Ours) Result

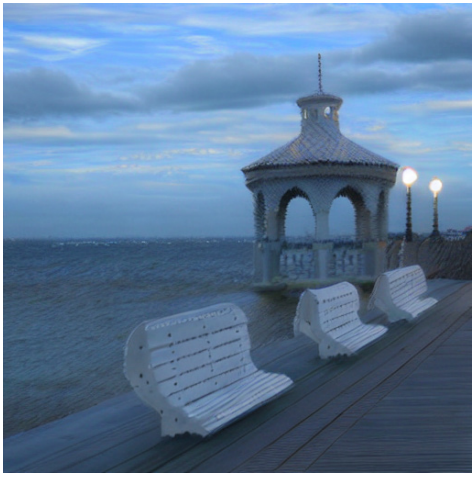
Figure 5: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image



(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 6: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 7: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 8: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 9: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 10: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

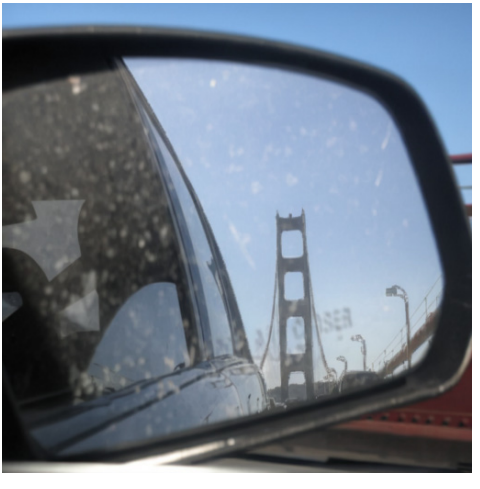
Figure 11: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

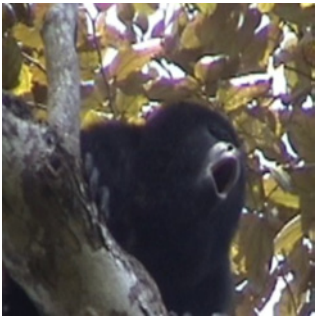
(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result



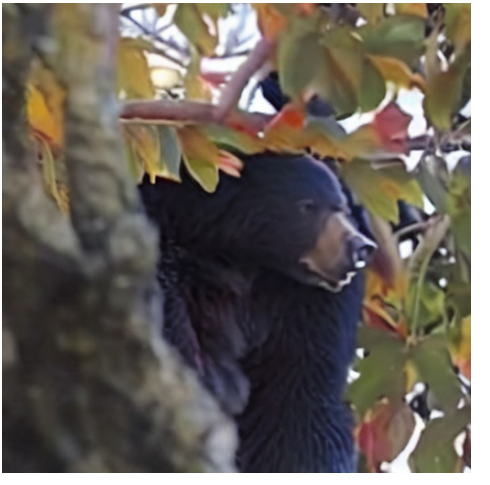
(e) VisualSplit (Ours) Result

Figure 12: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

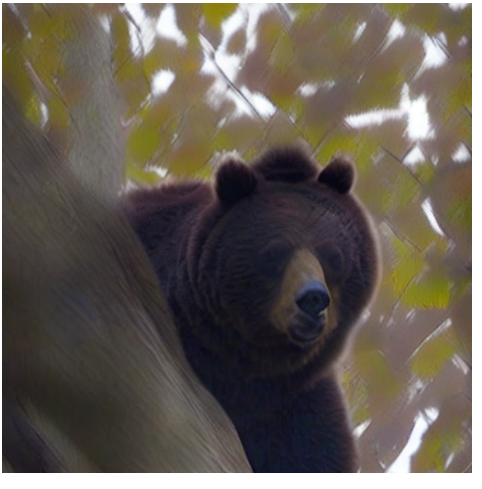


(a) Original Image

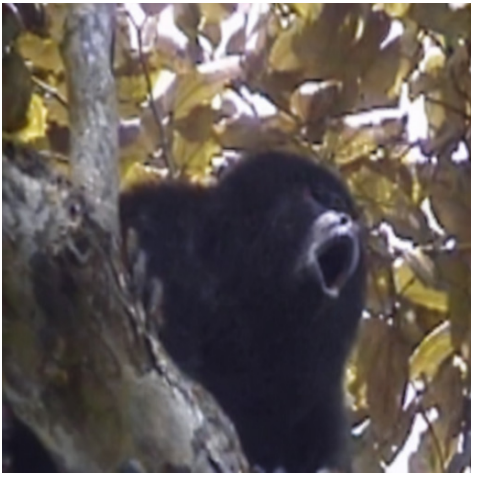
(b) ControlNet [14] w/ prompt Result



(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result



(e) VisualSplit (Ours) Result

Figure 13: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

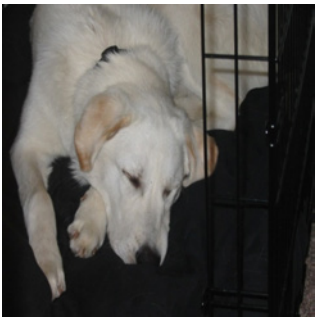
(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

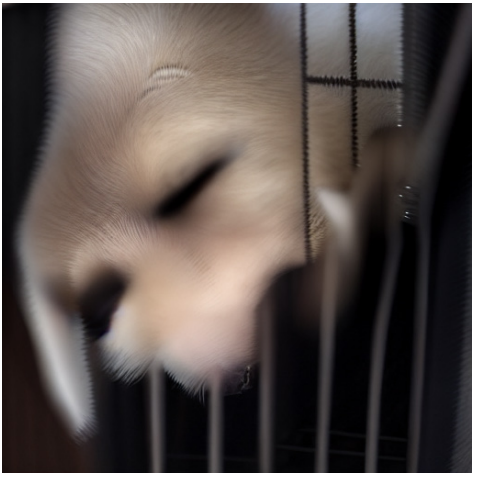
(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 14: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

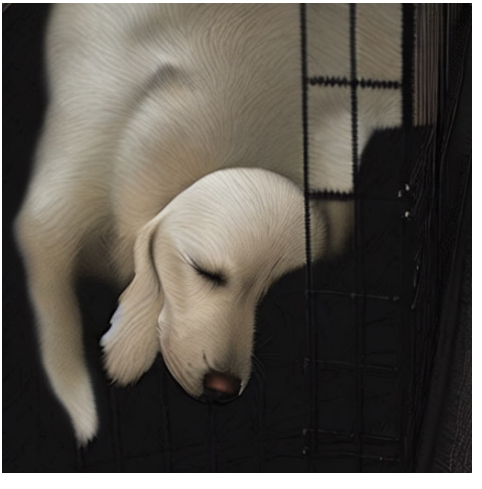


(a) Original Image

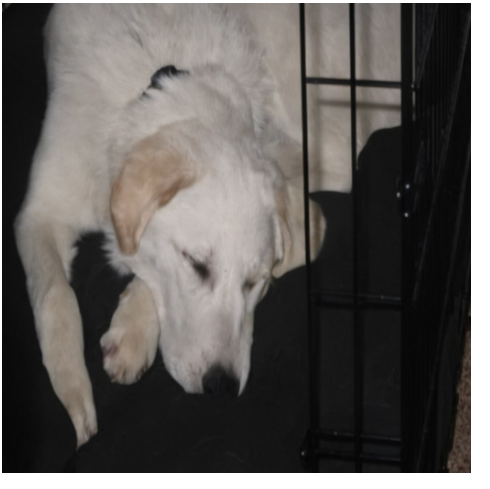


(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result



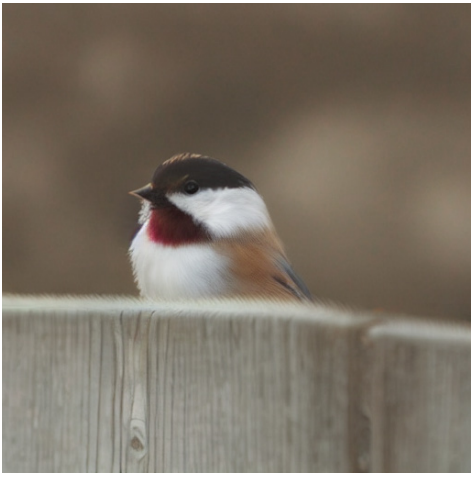
(e) VisualSplit (Ours) Result

Figure 15: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

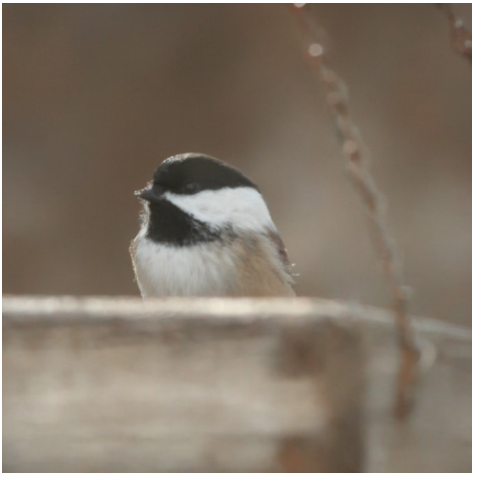
(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result



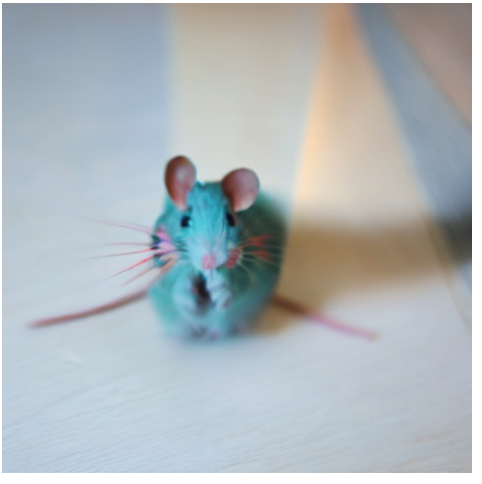
(d) ControlNet++ [7] w/ Prompt Result



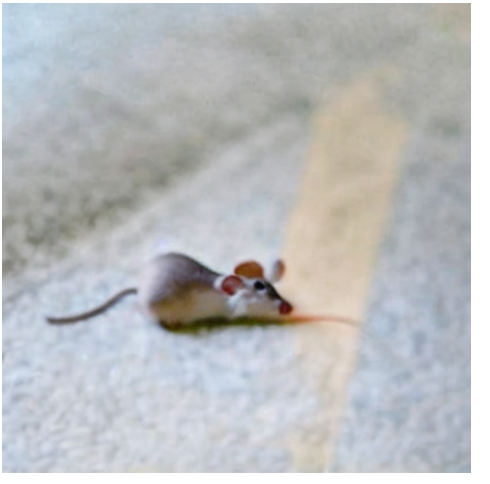
(e) VisualSplit (Ours) Result

Figure 16: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

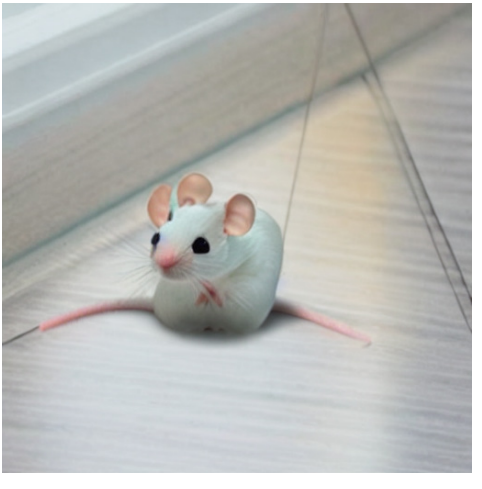
(a) Original Image



(b) ControlNet [14] w/ prompt Result



(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 17: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 18: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 19: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 20: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 21: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 22: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

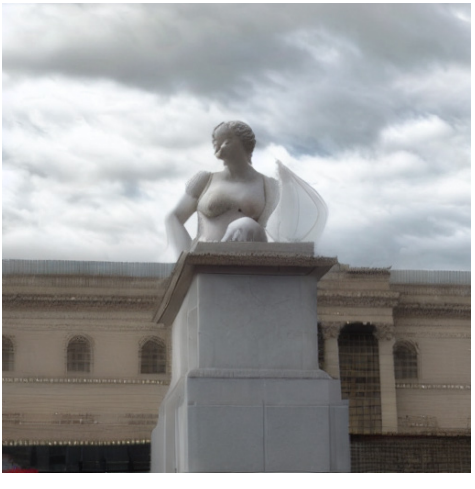
(e) VisualSplit (Ours) Result

Figure 23: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

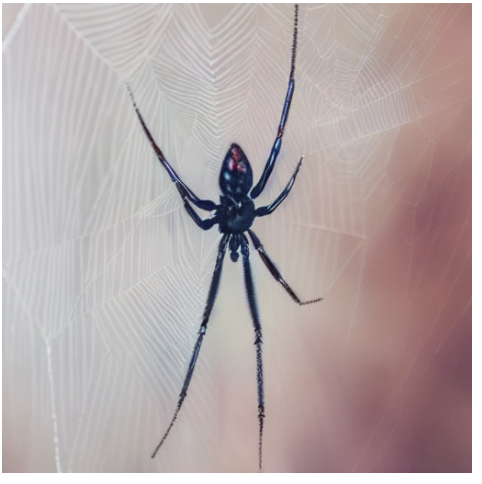


(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

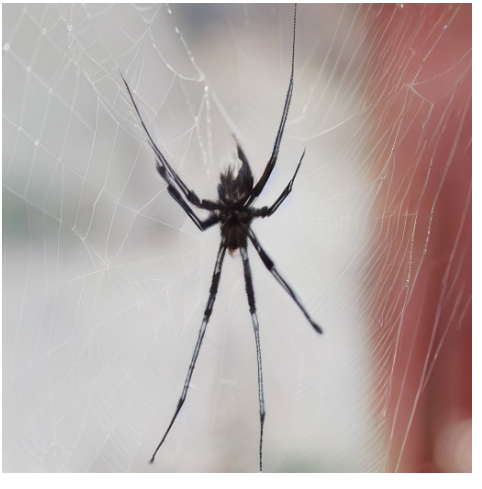
Figure 24: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

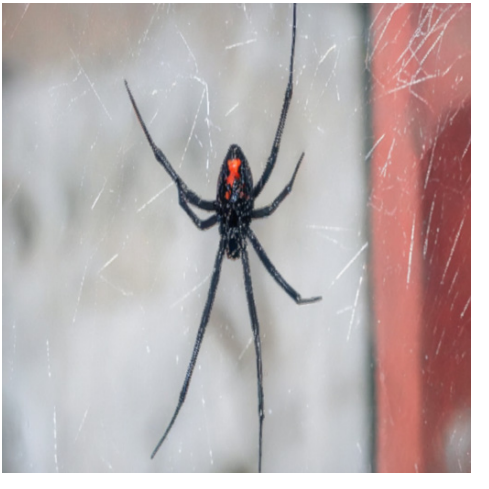


(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [1] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result



(e) VisualSplit (Ours) Result

Figure 25: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 26: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

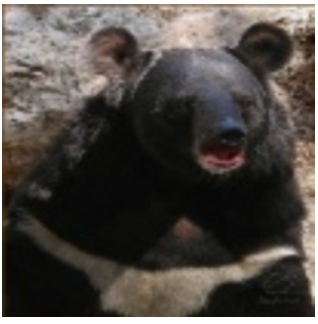
(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [1] w/ Prompt Result

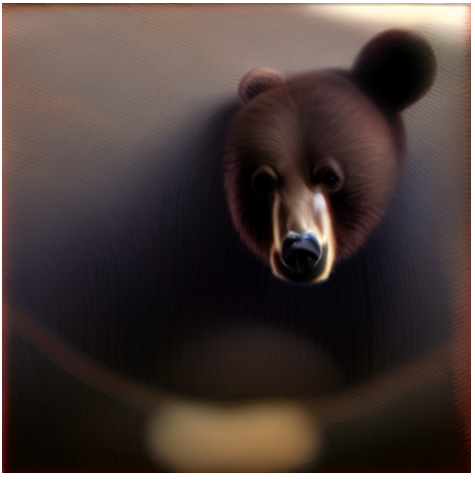
(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 27: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

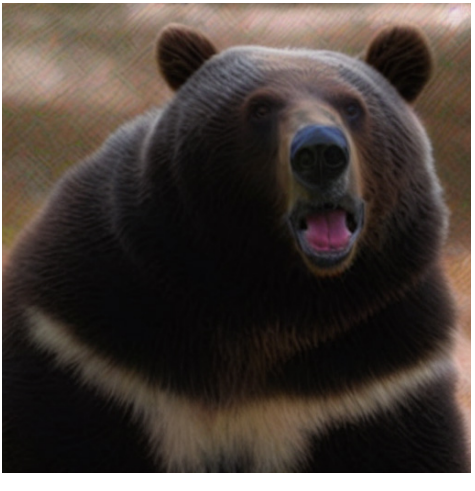


(a) Original Image



(b) ControlNet [14] w/ prompt Result

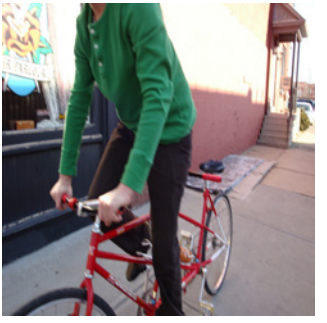
(c) T2I Adapter [11] w/ Prompt Result



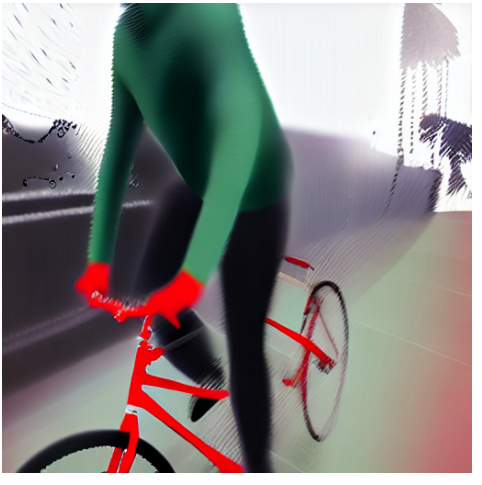
(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

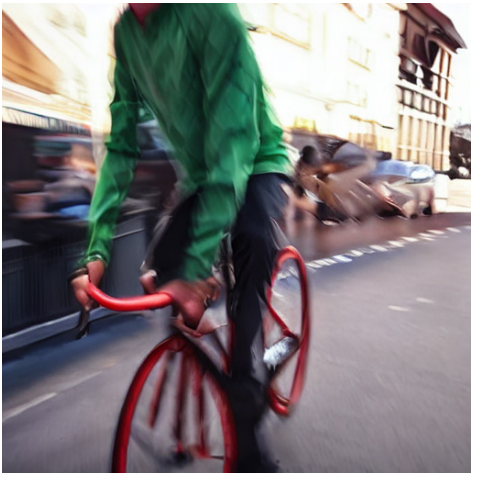
Figure 28: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.



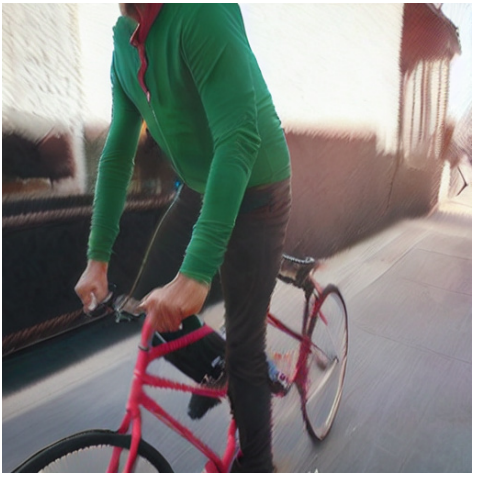
(a) Original Image



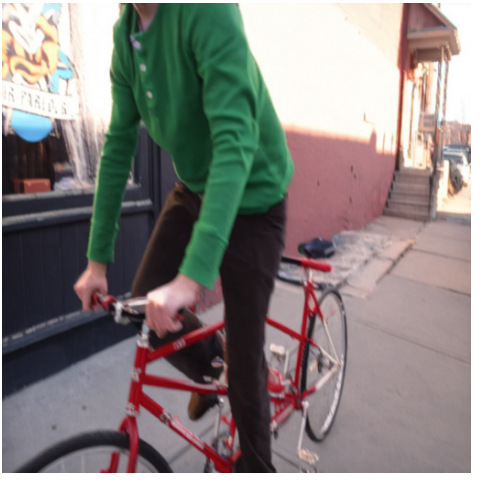
(b) ControlNet [14] w/ prompt Result



(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result



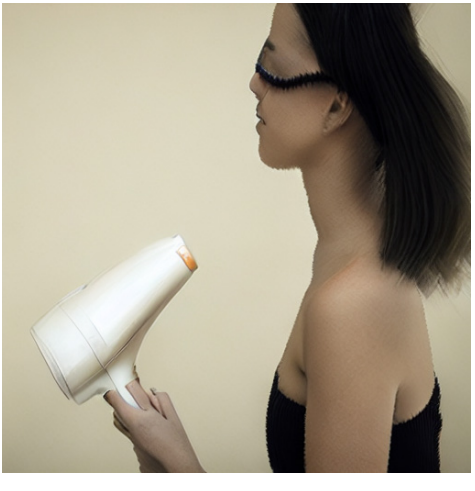
(e) VisualSplit (Ours) Result

Figure 29: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 30: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 31: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

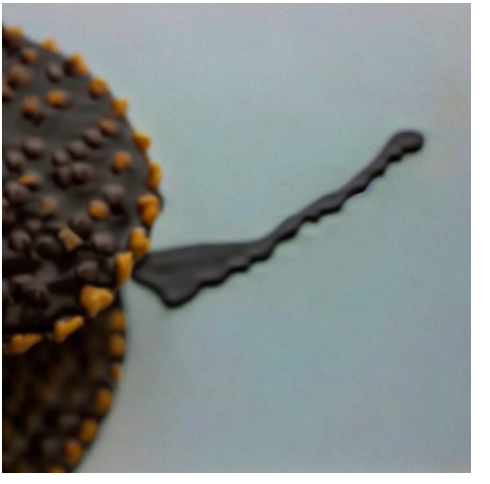
(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

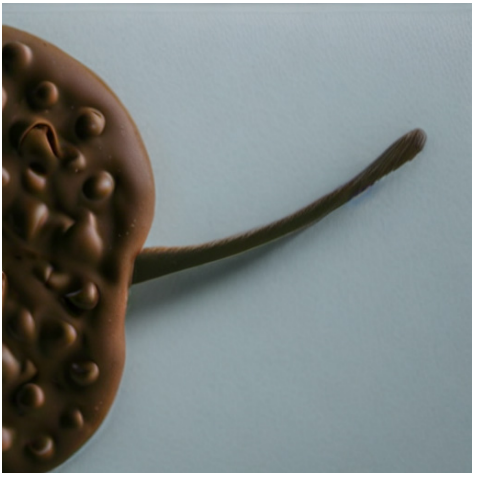
Figure 32: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result



(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 33: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 34: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

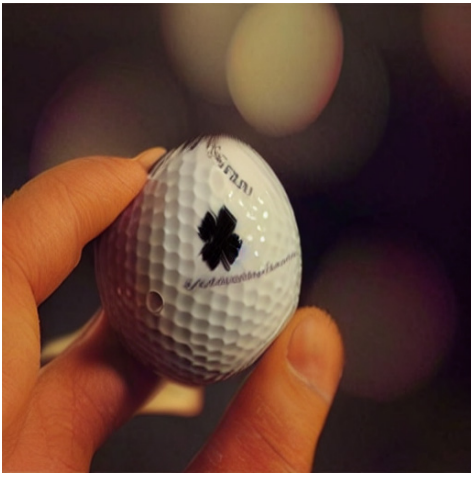
(e) VisualSplit (Ours) Result

Figure 35: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 36: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 37: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

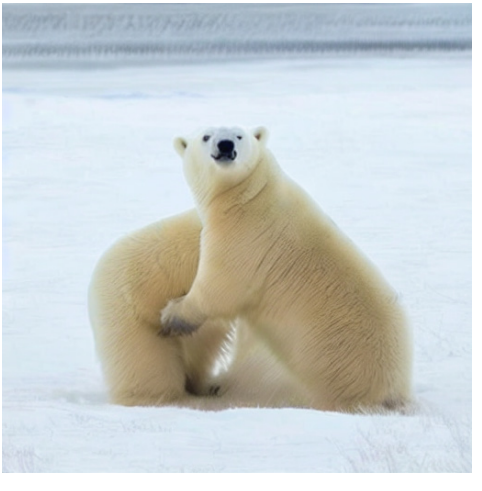
(e) VisualSplit (Ours) Result

Figure 38: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

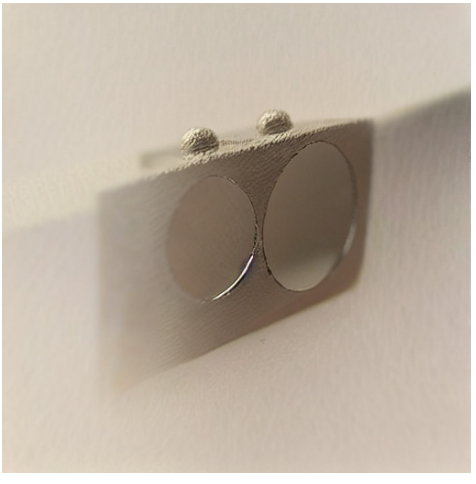


(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

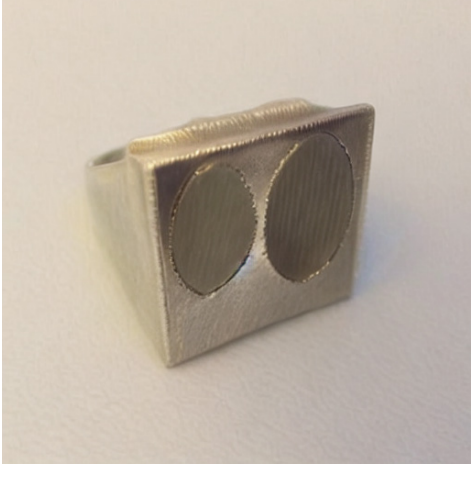
Figure 39: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image



(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 40: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [1] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 41: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

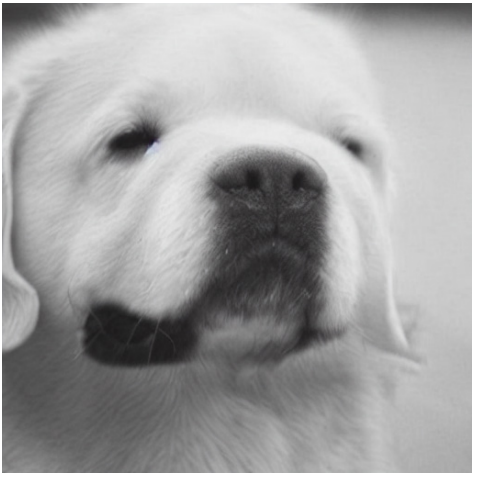
(e) VisualSplit (Ours) Result

Figure 42: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

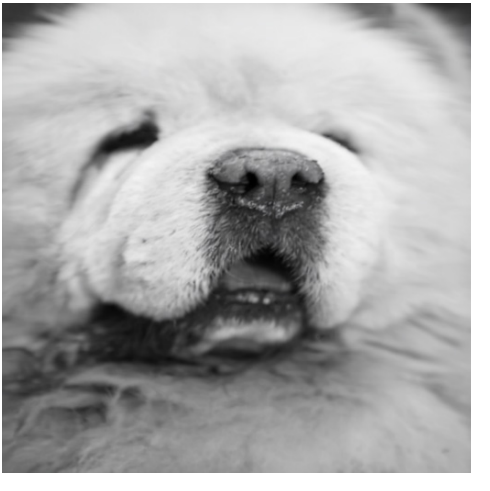
(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result



(e) VisualSplit (Ours) Result

Figure 43: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 44: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

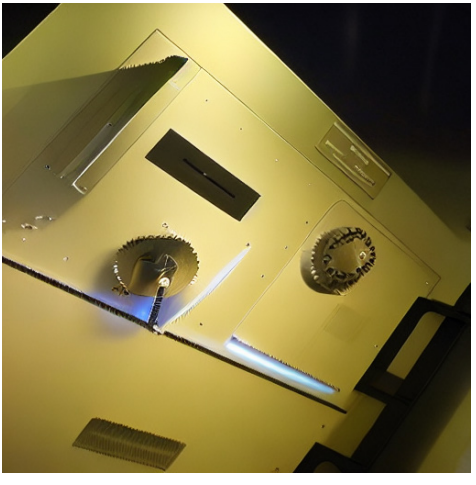
(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 45: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image



(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

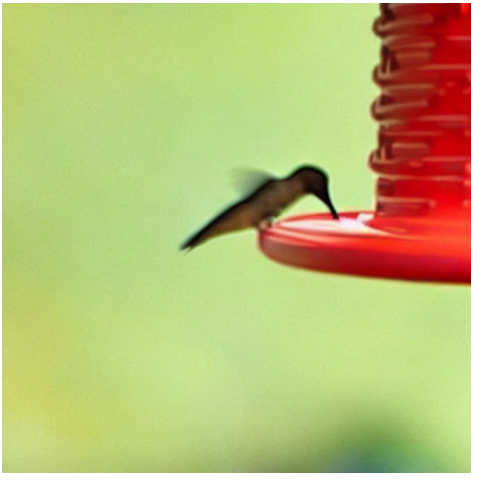
(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 46: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result



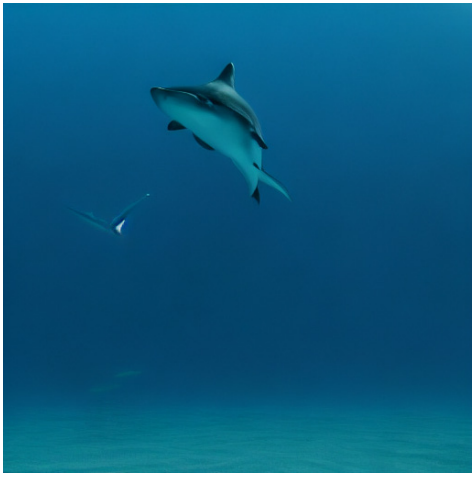
(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 47: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image



(b) ControlNet [14] w/ prompt Result

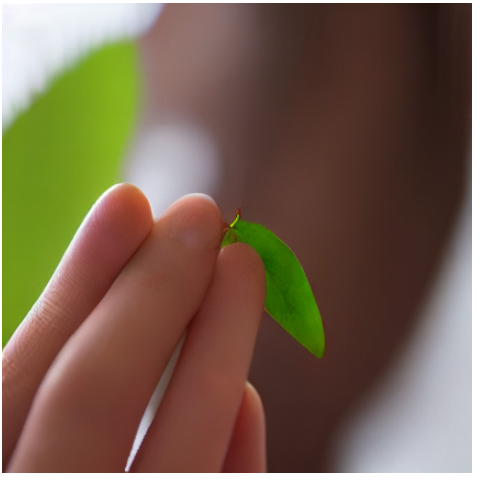
(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

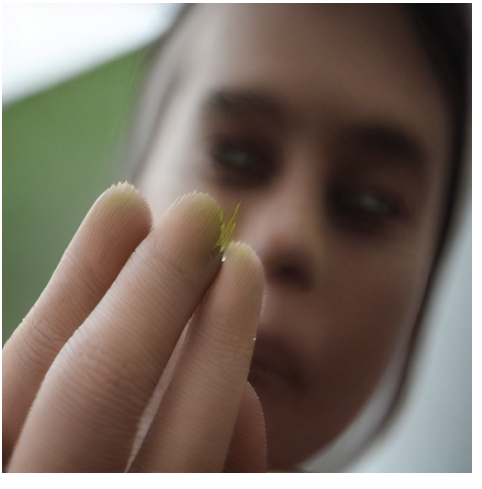
Figure 48: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image



(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

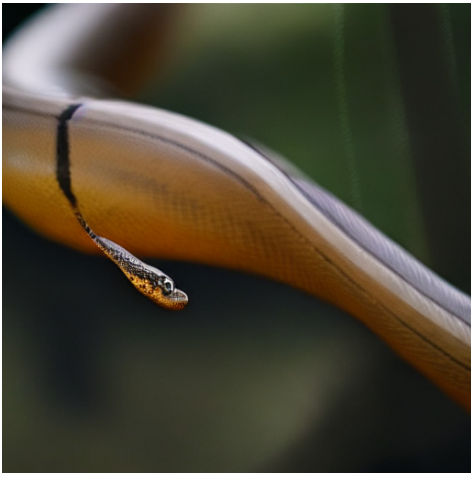


(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

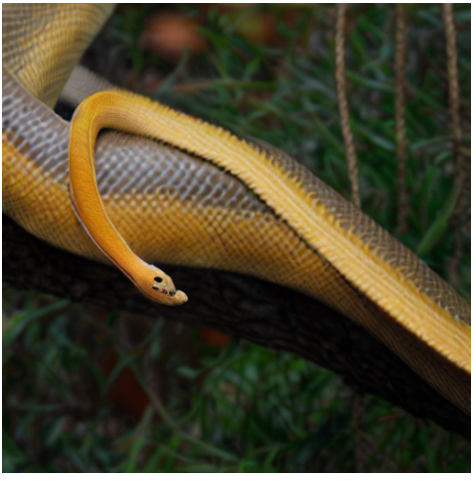
Figure 49: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image



(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 50: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 51: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

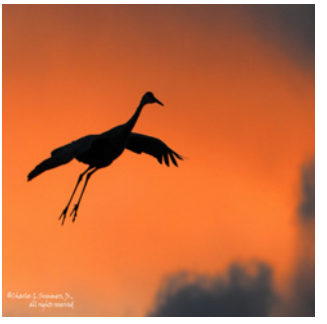
(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

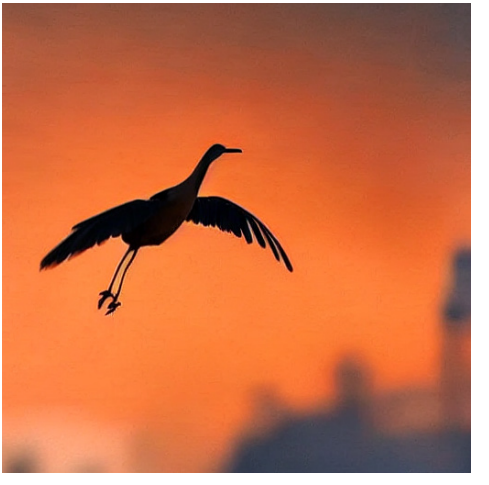
(e) VisualSplit (Ours) Result

Figure 52: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

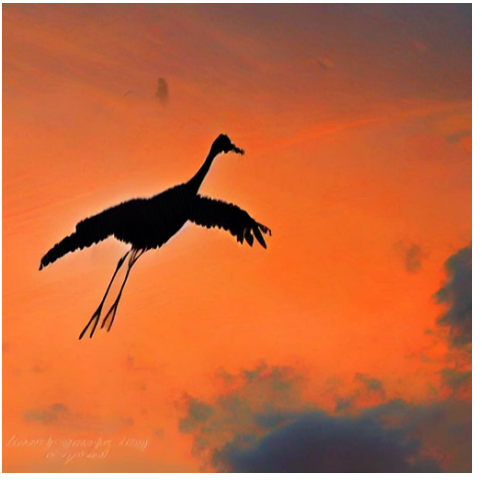


(a) Original Image

(b) ControlNet [14] w/ prompt Result



(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 53: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

(e) VisualSplit (Ours) Result

Figure 54: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result

(d) ControlNet++ [7] w/ Prompt Result

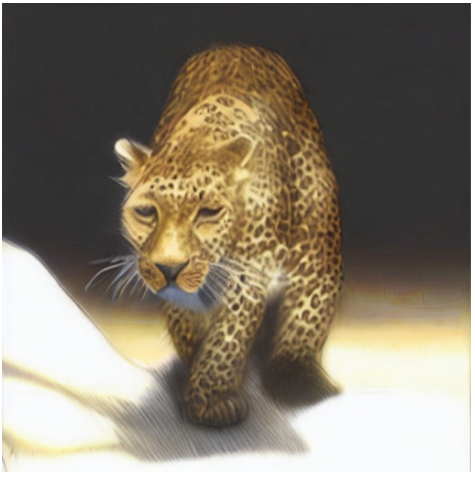
(e) VisualSplit (Ours) Result

Figure 55: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

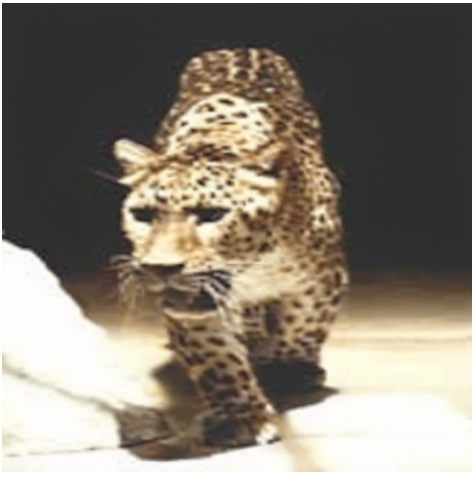
(a) Original Image

(b) ControlNet [14] w/ prompt Result

(c) T2I Adapter [11] w/ Prompt Result



(d) ControlNet++ [7] w/ Prompt Result

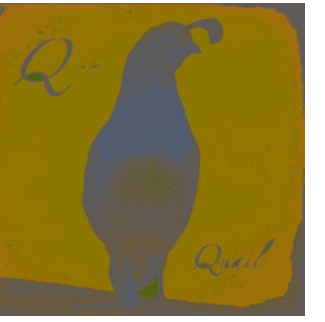


(e) VisualSplit (Ours) Result

Figure 56: Comparison of colour restoration results from different models using descriptors as input with Stable Diffusion 1.5.

(a) Original Image

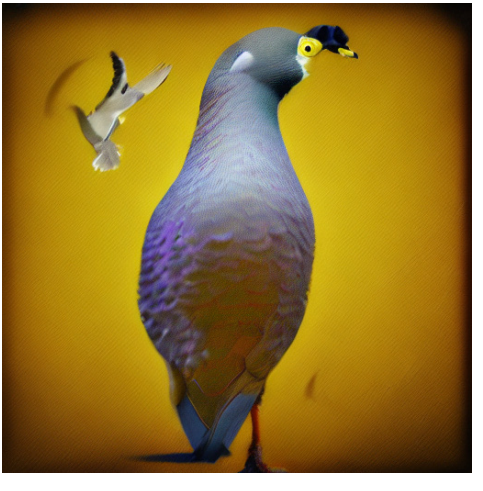
(b) Original Colour Map



(c) Edited Colour Map

(d) ControlNet [14] w/ prompt Result

(e) T2I Adapter [11] w/ Prompt Result



(f) ControlNet++ [7] w/ Prompt Result

(g) VisualSplit (Ours) Result

Figure 57: Comparison of colour editing results using modified segmented colour maps as input with Stable Diffusion 1.5 with different guidance methods. The prompts generated by BLIP with edited colour keywords.

(a) Original Image

(b) Original Colour Map

(c) Edited Colour Map

(d) ControlNet [14] w/ prompt Result

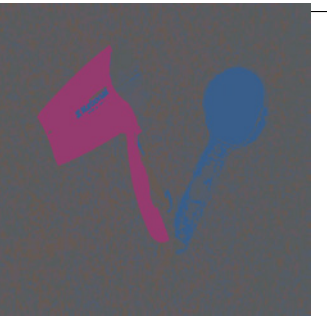
(e) T2I Adapter [11] w/ Prompt Result

(f) ControlNet++ [7] w/ Prompt Result

(g) VisualSplit (Ours) Result

Figure 58: Comparison of colour editing results using modified segmented colour maps as input with Stable Diffusion 1.5 with different guidance methods. The prompts generated by BLIP with edited colour keywords.

(a) Original Image



(b) Original Colour Map

(c) Edited Colour Map

(d) ControlNet [14] w/ prompt Result

(e) T2I Adapter [11] w/ Prompt Result

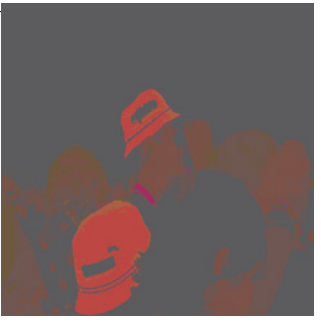
(f) ControlNet++ [7] w/ Prompt Result

(g) VisualSplit (Ours) Result

Figure 59: Comparison of colour editing results using modified segmented colour maps as input with Stable Diffusion 1.5 with different guidance methods. The prompts generated by BLIP with edited colour keywords.

(a) Original Image

(b) Original Colour Map

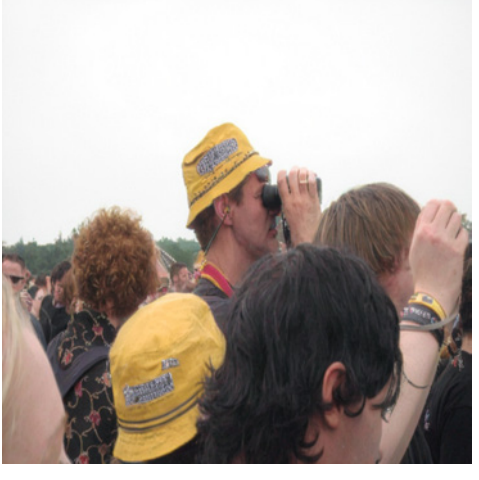


(c) Edited Colour Map

(d) ControlNet [14] w/ prompt Result

(e) T2I Adapter [11] w/ Prompt Result

(f) ControlNet++ [7] w/ Prompt Result



(g) VisualSplit (Ours) Result

Figure 60: Comparison of colour editing results using modified segmented colour maps as input with Stable Diffusion 1.5 with different guidance methods. The prompts generated by BLIP with edited colour keywords.

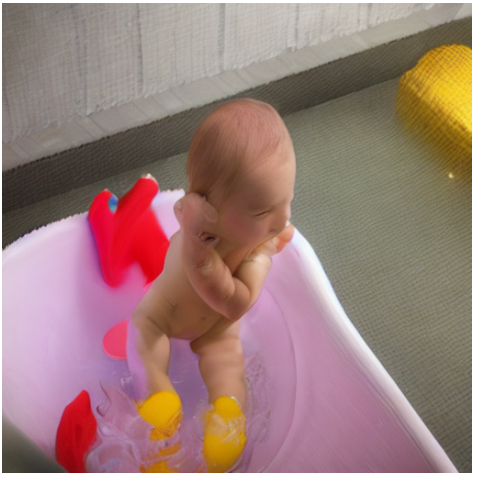
(a) Original Image

(b) Original Colour Map

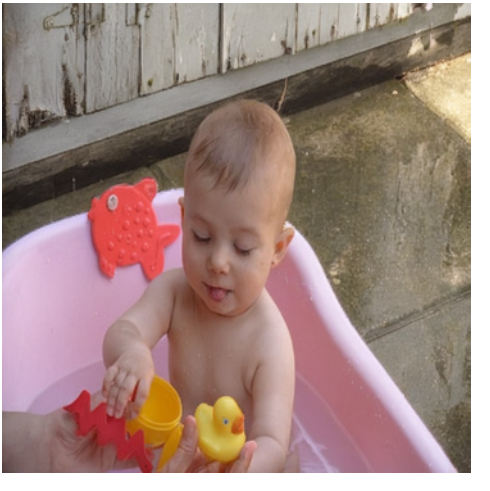
(c) Edited Colour Map

(d) ControlNet [14] w/ prompt Result

(e) T2I Adapter [11] w/ Prompt Result



(f) ControlNet++ [7] w/ Prompt Result



(g) VisualSplit (Ours) Result

Figure 61: Comparison of colour editing results using modified segmented colour maps as input with Stable Diffusion 1.5 with different guidance methods. The prompts generated by BLIP with edited colour keywords.