FastCLIPstyler: Optimisation-free Text-based Image Style Transfer Using Style Representations (Supplementary Materials)

Ananda Padhmanabhan Suresh* Sanjana Jain* Pavit Noinongyao Ankush Ganguly
Ukrit Watchareeruetai Aubin Samacoits

Sertis Vision Lab
597/5 Sukhumvit Road, Watthana, Bangkok, 10110, Thailand
{asure,sjain,ponio,agang,uwatc,asama}@sertiscorp.com
*Both authors contributed equally to this work

1. Model implementation details

1.1. Model architecture

In our research, we propose two main text-based image style transfer frameworks, namely FastCLIPstyler and EdgeCLIPstyler. While FastCLIPstyler performs stylisation in a single forward pass with state-of-the-art stylisation quality, EdgeCLIPstyler supports stylisation on edge devices with a slight trade-off in stylisation quality compared to FastCLIPstyler. The architecture diagrams of FastCLIPstlyer and EdgeCLIPstyler can be seen in Fig. 1(a) and Fig. 1 (b), respectively.

During the dataset generation phase, FastCLIPstyler and EdgeCLIPstyler optimise their respective text-style prediction networks for each prompt, to obtain the corresponding style embedding. To achieve this, both frameworks use the CLIP text and image embedder to compute the CLIP loss for optimisation. In order to generate the input text embedding for the text-style prediction network, FastCLIPstyler uses the CLIP text embedder, while the EdgeCLIPstyler framework uses the Sentence-BERT text embedder [6]. During inference, EdgeCLIPstyler replaces FastCLIPstyler's CLIP text embedder with Sentence-BERT, eliminating the need for CLIP during inference and resulting in faster speed and compatibility with edge devices. It is to be noted that the CLIP image embedder is only required for loss computation, and is not utilised during inference for either framework.

1.2. Text-style prediction network

The text-style prediction network is a simple fully-connected feed-forward network that takes a text embedding as input and converts it into a 100-dimensional style embedding.

For FastCLIPstyler, a 512-dimensional text embedding is extracted from CLIP and passed as input to the text-style prediction network. The text-style prediction network comprises four hidden layers of 256, 256, 150, and 150 nodes, each with the Leaky ReLU [5] activation function with a negative slope of 0.2 as the activation function. The final layer uses the tanh activation function to normalise the style representations between -1 and 1 since this is the range of values of style representation in the Ghiasi model [4].

On the other hand, EdgeCLIPstyler adopts the Sentence-BERT paraphrase-albert-v2 text embedder [6] to extract a 768-dimensional text embedding, which is passed as input to the text-style prediction network. In order to accommodate the larger 768-sized text embedding, we have made modifications to the text-style prediction network to accept 768-dimensional as input. The text-style prediction network, in this case, consists of five hidden layers with 512, 256, 256, 150, and 150 nodes, respectively, while adopting the same activation function setup as FastCLIPstyler.

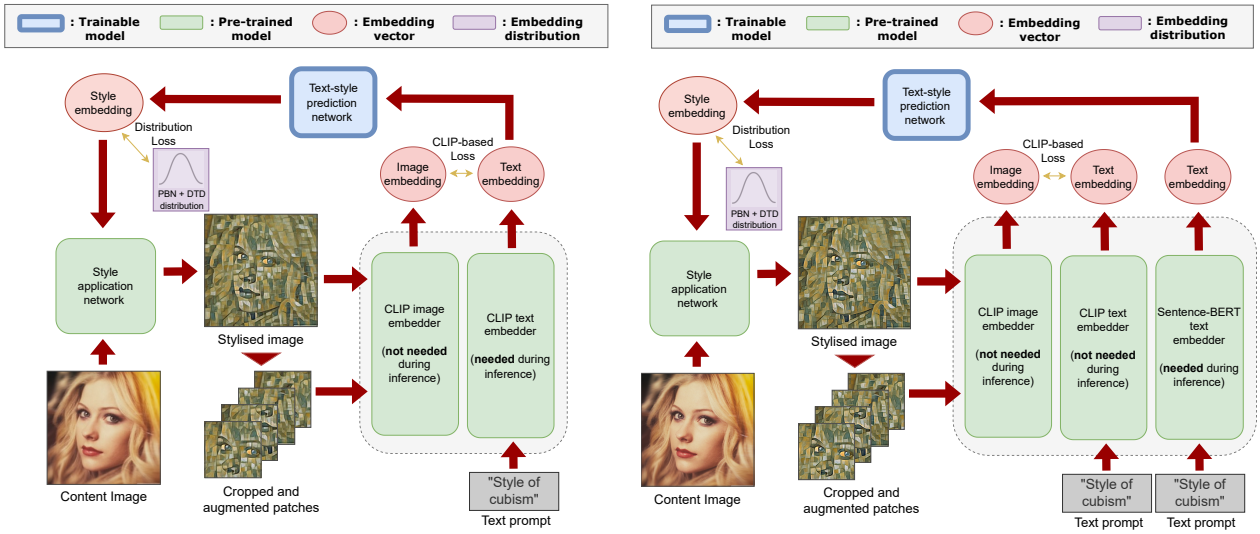
1.3. Model training and hyperparameters

1.3.1 Dataset generation

A key step in the building of the FastCLIPstyler and EdgeCLIPstlyer models is the training of the text-style prediction network. In order to train this network, we start by fitting the network for each text embedding individually and extracting the corresponding style embedding. This process is described by the following equation:

$$e_i^{style} = f(e_i^{text}, \theta_i), \tag{1}$$

where e_i^{style} is the style embedding, $f(\cdot)$ the text-style prediction network, θ_i is a parameter fitted for each prompt's CLIP text embedding e_i^{text}.



(a) FastCLIPstyler

(b) EdgeCLIPstyler

Figure 1. Architectural diagram of our two proposed approaches.

1.3.2 Generalised model training

These style embeddings, obtained from the dataset generation step, are then used as 'labels' for the model training stage, where the text-style prediction network is trained to map from the text embedding to its corresponding 'label' style embedding using the standard mean-squared error (MSE) loss given by:

$$MSE = \frac{1}{N}\sum_{i=1}^{N} \|e_i^{style} - \hat{e}_i^{style}\|^2, \tag{2}$$

where N is the number of text prompts in the dataset, $\hat{e}_i^{style}$ is the predicted style embedding at any particular stage in the training procedure, and e_i^{style} is the pre-computed style embedding from Eq. 1.

The various hyper-parameters used to train the models and the time taken are enlisted in Tab. 1.

2. Datasets

2.1. Keyword-combination based prompts

To train our model, we constructed a dataset of queries by combining a list of keywords that included 44 different shades of colours, 21 textures, 5 art styles, and descriptions of 34 real-world objects with distinguishable textures. The list of keywords can be seen in Tab. 2. Various random combinations such as colour-texture, colour-art, colour-object, art-texture, and art-object were created to generate the dataset of 4,302 prompts. Examples of the keyword combination prompts can be seen in Tab. 3. We followed specific strategies to preserve the legibility of this final list of generated prompts. For instance, we appended the word 'colour' at the end of each of the 44 shades. This led to the removal of ambiguity for colours such as 'salmon', 'seashell', 'chocolate', etc. In the case of textures, we selected 21 different textures from the DTD dataset by filtering out textures that seemed visually repetitive. We randomly combined these textures with the set of colours and text descriptions of real-world objects such as 'stone wall', 'cloud', 'fire' to name a few. Some examples of such combinations include 'maroon crosshatched', 'yellow freckled', 'black wrinkled', etc. Additionally, the dataset consists of five different art styles, namely 'acrylic', 'oil painting', 'mosaic', 'cubism', and 'monet', which too were randomly combined with the set of colours and textures. The resulting combinations include prompts such as 'pink oil painting', 'mosaic cloud', and 'studded cubism', among others. Furthermore, we randomly augmented the final list of prompts with leading phrases such as 'style of', 'a pixelated photo of', 'a blurry photo of', 'a sketch of a', and 'a black and white photo', as well as with trailing phrases such as 'style' and 'style painting'. Examples after this augmentation include 'style of blue colour', 'a blurry photo of white wool', 'a pixelated photo of cyan stone wall', and many more. Experimental results show that our network generalises while being trained on this dataset and can generate stylised images with unseen queries of quality comparable to the current state-of-the-art.

	Learning rate	Epochs	λ_{dir}	λ_{patch}	λ_{dis}	Time taken
Data generation	1×10^{-3}	150	500	1500	0.5	14.5 hours
Model training	1×10^{-5}	200	-	-	-	3 minutes

Table 1. Various hyperparameters used to train the different experiment runs and the time taken.

Categories	Keywords
Colors	White, Black, Green, Yellow, Pink, Purple, Blue, Orange, Red, Violet, Silver, Gray, Maroon, Teal, Aqua, Beige, Brown, Crimson, Cyan, Gold, Greenyellow, Hotpink, Khaki, Magenta, Turquoise, Fuchsia, Salmon, Seashell, Chocolate, Peru, Whitesmoke, Honeydew, Rosybrown, Saddlebrown, Seagreen, Slategray, Steelblue, Indianred, Olive, Lime, Navy, Lavender, Indigo, Ivory
Objects	Wool, Color pencil, Pencil, Brush, Color brush, Crystals, color, Crystals, Color painting, Crayon, Lines, Watercolor, Stone wall, Cloud, Underwater, Fire, Metal, Lightning, Wave, Flames, Leafy, Grassy, Darkness, Wooden, Snow, Iceberg, Cartoon, Comic, Squares, Lava, Vines, Magma, Desert sand, Water waves, Lace
Art styles	Acrylic, Oil Painting, Mosaic, Cubism, Monet
Textures	Crystalline, Cracked, Crosshatched, Fibrous, Freckled, Grid, Honeycombed, Meshed, Perforated, Porous, Scaly, Smeared, Studded, Swirly, Veined, Waffled, Woven, Wrinkled, Pleated, Sprinkled, Knitted

Table 2. Keywords used for forming the combination prompts dataset

2.2. ChatGPT based prompts

CLVA's [3] research effectively illustrated the potential of the ArtEmis dataset [1] for enhancing the training of style transfer systems. While integrating the ArtEmis dataset into our style transfer model training, we identified a mismatch between the dataset's content-descriptive prompts and our requirement for style-focused prompts. ArtEmis prompts, though rich and varied, contains a large number of annotations that refer to visible contents rather than capturing the essence of artistic style. To rectify this, we engaged ChatGPT [2] to adapt and generate prompts that were more aligned with our model's objectives. We selected representative prompts from ArtEmis that subtly suggested style and provided ChatGPT with guidelines to craft prompts that foreground elements of style such as texture, colour scheme, and compositional flow. The utilisation of ChatGPT for this task resulted in an enriched set of 1,500 prompts that more closely cater to the model's need for stylistic abstraction. These prompts avoid direct content representation, fostering a training environment that emphasises the stylistic interpretation rather than the literal depiction. The effectiveness of this tailored dataset is evident in the improved model performance and is exemplified by the curated examples in Tab. 3, which demonstrate the prompts' adherence to the model's style-centric learning paradigm.

3. Embedding space mapping

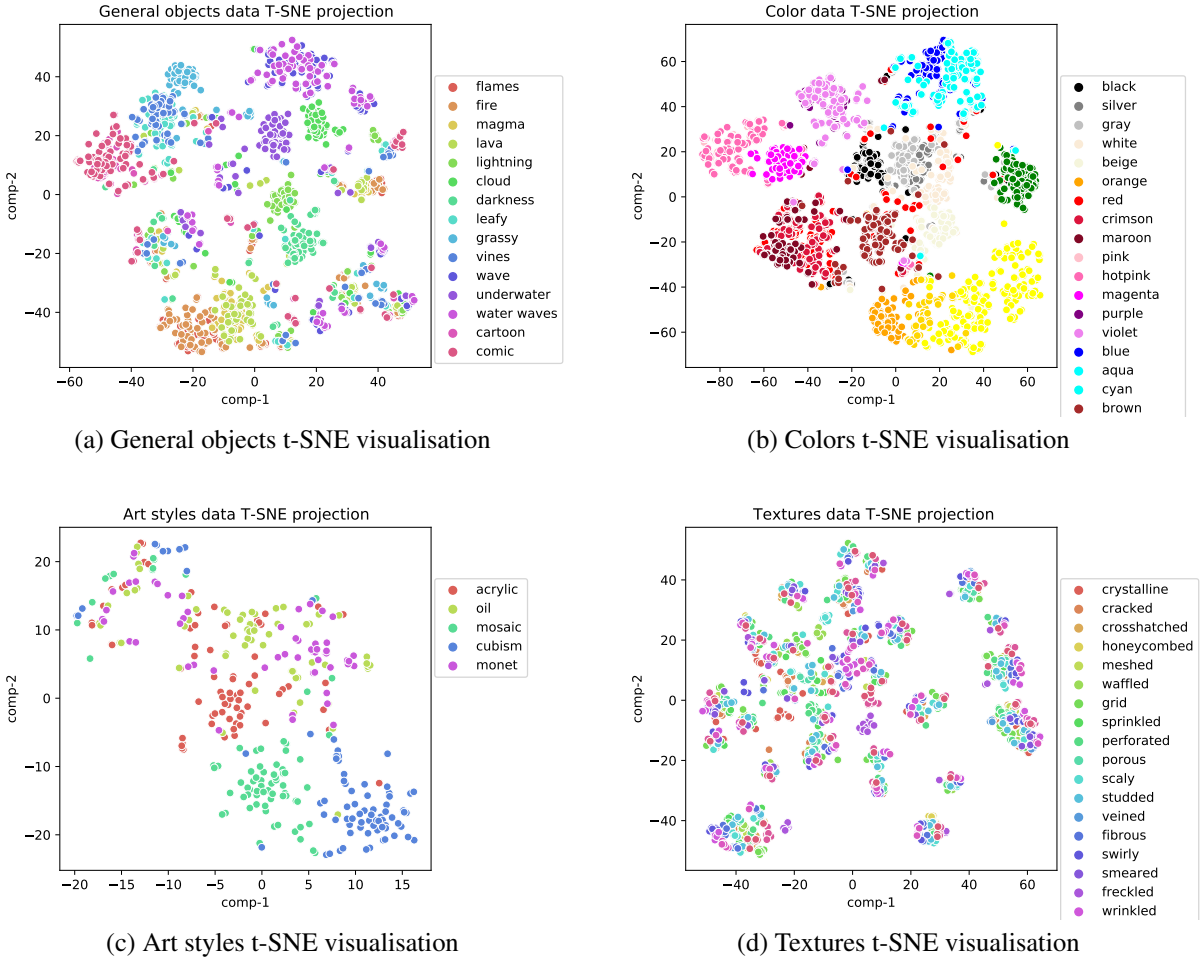
Ghiasi *et al*. [4] have successfully demonstrated that the embedding space of their style transfer network captures semantic information about styles. As we adopt their style embedding space to fit our text-style prediction network, we verify that our prediction network is able to preserve the semantic information upon mapping from the text embeddings.

To do so, we generate the text embeddings and corresponding style embeddings for various text prompts using our generalised text-style prediction network. As the style embedding vectors are 100-dimensional, for visualisation purposes, we perform dimensionality reduction using t-SNE [7]. Figure 2 illustrates the two-dimensional t-SNE plot of the style embeddings obtained by passing various combinations of generated text prompts through our text-style prediction network. As can be seen, our prediction network successfully maps semantically similar queries closer together in the dimensionally-reduced style embedding space.

For Fig. 2(a), we built a dataset of 1,516 text prompts comprising 15 keywords {fire, flames, magma, lava, leafy, vines, grassy, wave, water waves, underwater, cartoon, comic, lightning, cloud, darkness} combined with various colours and textures, forming prompts like 'a blurry photo of red underwater', 'cracked leafy style', and 'a pixelated

Categories	Prompts
Keyword combination prompts	White color, Blue color pencil, Desert sand, Style of maroon wool, Orange cracked, Aqua colour style Mosaic style painting, Cracked acrylic painting, etc.
ChatGPT generated prompts	Ocean waves and serene calmness, Glittering stars in the night sky, A dreamlike landscape bathed in the soft light evoking feelings of tranquillity

Table 3. Samples of keyword combination prompts and ChatGPT generated prompts.



(a) General objects t-SNE visualisation

(b) Colors t-SNE visualisation

(c) Art styles t-SNE visualisation

(d) Textures t-SNE visualisation

Figure 2. The t-SNE visualisation of style embedding space upon mapping from the text prompts.

photo of magma'. We also grouped these keywords together into semantically meaningful sets, including {fire, flames, magma, lava}, {leafy, vines, grassy}, {wave, water waves, underwater}, {cartoon, comic} and {lightning, cloud, darkness}. We demonstrate that for prompts having similar semantic meanings, their style embeddings lie very close together in the embedding space. For example, prompts containing 'fire' and 'flames' or 'magma' and 'lava' almost overlap in the embedding space. We see similar behaviour for all the above-mentioned semantically meaningful sets. Due to the complexity of these prompts formed from pairing keywords with colours and textures, we witness some outliers lying outside the main distribution. However, in most cases, they preserve a similar structure of semantically

meaningful queries.

For the visualisation of colour-based prompts (Fig. 2(b)), we generated 2,968 prompts containing 22 colours combined with various general objects, art styles, and textures. We can see similar colour-based prompts lie closer in the embedding space. As for art style-based prompts (Fig. 2(c)), we generate 209 prompts for five art styles {acrylic, monet, oil, cubism, and mosaic}, combined with various colours and textures. We can see art styles such as 'cubism' and 'monet', which have a similar style, lying close in the embedding space. Additionally, 'acrylic', 'oil', and 'monet', again with similar styles, lie close in the embedding space. We also visualise 1,516 texture-based prompts (Fig. 2(d)), comprising 21 textures combined with various colours and objects. While the plot is scattered due to combining with various other well-preserved colours and objects, it seems to preserve semantic meaning within these scattered distributions.

4. General model performance

We share additional stylisation results of our models on various general prompts, art styles, and textures. The results for FastCLIPstyler can be seen in Fig. 3, Fig. 4, and Fig. 5. The results for EdgeCLIPstyler can be seen in Fig. 6.

5. Quantitative evaluation - Human evaluation

Human evaluation was done by means of an online form survey. We created four distinct forms, each with 20 questions. A total of 75 participants were randomly given one of these forms. All forms start with giving information on how style transfer works and what the desirable and undesirable properties of a stylised image are with visual examples. Sample images of the form are shown in Fig. 7. Each section of the form shows a style prompt and displays a content image and its corresponding stylised images generated by different models. The participants were then asked to rate from 1 to 5 the quality of each image. The final scores for each model are then calculated as the mean of the scores from the form responses.

The prompts and associated content images were chosen to ensure a fair representation. We randomly select prompts from various databases, such as ArtEmis and DTD, which were utilised for model evaluation in the original research by CLVA. Similar methodologies were employed for the selection of prompts from CLIPstyler and our own model evaluations. This strategy ensures an unbiased comparison across all models under review. For fairness purposes, the stylised images shown to the participants were not labelled with their corresponding models and were arranged randomly to reduce potential order bias.

For text prompt selection, we randomly selected a subset of text prompts generated for training the text-style prediction network. We additionally create a set of general prompts for human evaluation. For content images, we randomly selected different content images from our qualitative evaluation experiments. Then from a pool of final prompts, we split them into four evaluation forms, 20 prompts each. Additional user study result is shown in Fig. 8 as a box plot of rated scores from 1 to 5 for each of the method.

6. Additional comparison results

We compare the performance of our model to the state-of-the-art text-based image style transfer approaches, namely CLIPstyler and CLVA. Experimental results show that our FastCLIPstyler results are on par with the quality of these other approaches in general. While CLIPstyler produces impressive stylisation results due to the advantage of optimising the process at run-time, it tends to sometimes over-stylise images and introduce artefacts as well. Over-stylisation in this context refers to the case when the style is applied too heavily, to the point where it overwhelms the content of the image or even adds artefacts (unwanted objects that represent content instead of style) onto the image. One reason for this could arise from the high flexibility in the architecture of CLIPstyler. The CNN that styles the image does not have a direct understanding of artistic paintings or specific art styles. Instead, it achieves this indirectly through the loss function defined by CLIP. As a result, the model has 'too much freedom' in editing the image, which can sometimes lead to undesired results.

Although CLVA can stylise images in a single forward pass, it falls short in terms of generalising to diverse prompts. Our FastCLIPstyler model provides stylisation results on par with these approaches and generalises to diverse prompts while being able to produce results in a single forward pass. Our EdgeCLIPstyler supports edge devices with a minor drop in performance. A qualitative comparison of our models with other recent state-of-the-art text-based image style transfer approaches is shown in Fig. 9.

7. Negative social impact

Text-based image style transfer has made it easy to manipulate an image with a style description. This is both beneficial and also can have negative social impacts. Since a reference style image is not required, users are able to manipulate any image using just words. This might have the highest negative impact on people who work in the graphical design area. If developed further, it could potentially replace those jobs in some cases that do not require sophisticated work. Also, we do not know the full capability of this model. It might be able to transform an image in unexpected ways that might cause harm to the user.

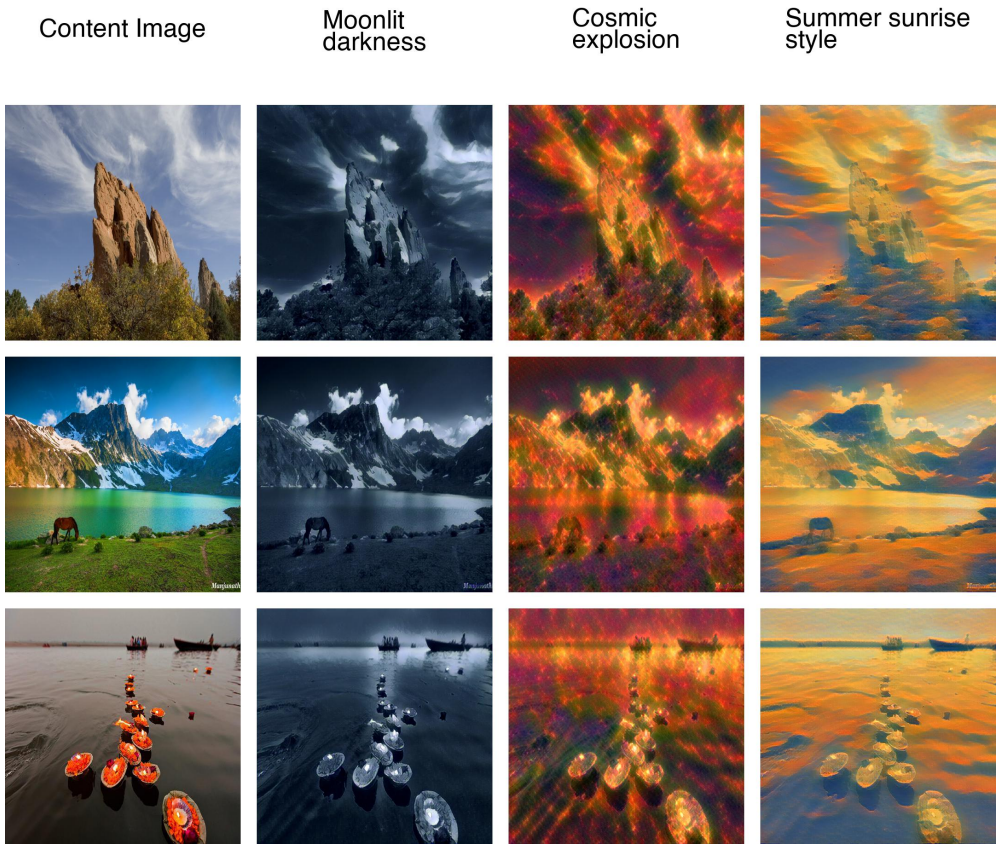


Figure 3. Performance of our FastCLIPstyler model on general style prompts.

References

[1] Panos Achlioptas, Maks Ovsjanikov, Kilichbek Haydarov, Mohamed Elhoseiny, and Leonidas Guibas. ArtEmis: Affective language for visual art. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11564–11574, 2021. 3

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. 3

[3] Tsu-Jui Fu, Xin Eric Wang, and William Yang Wang. Language-driven artistic style transfer. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVI*, pages 717–734. Springer, 2022. 3

[4] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. In *British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4-7, 2017*. BMVA Press, 2017. 1, 3

[5] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, page 3, 2013. 1

[6] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. 1

[7] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. 3

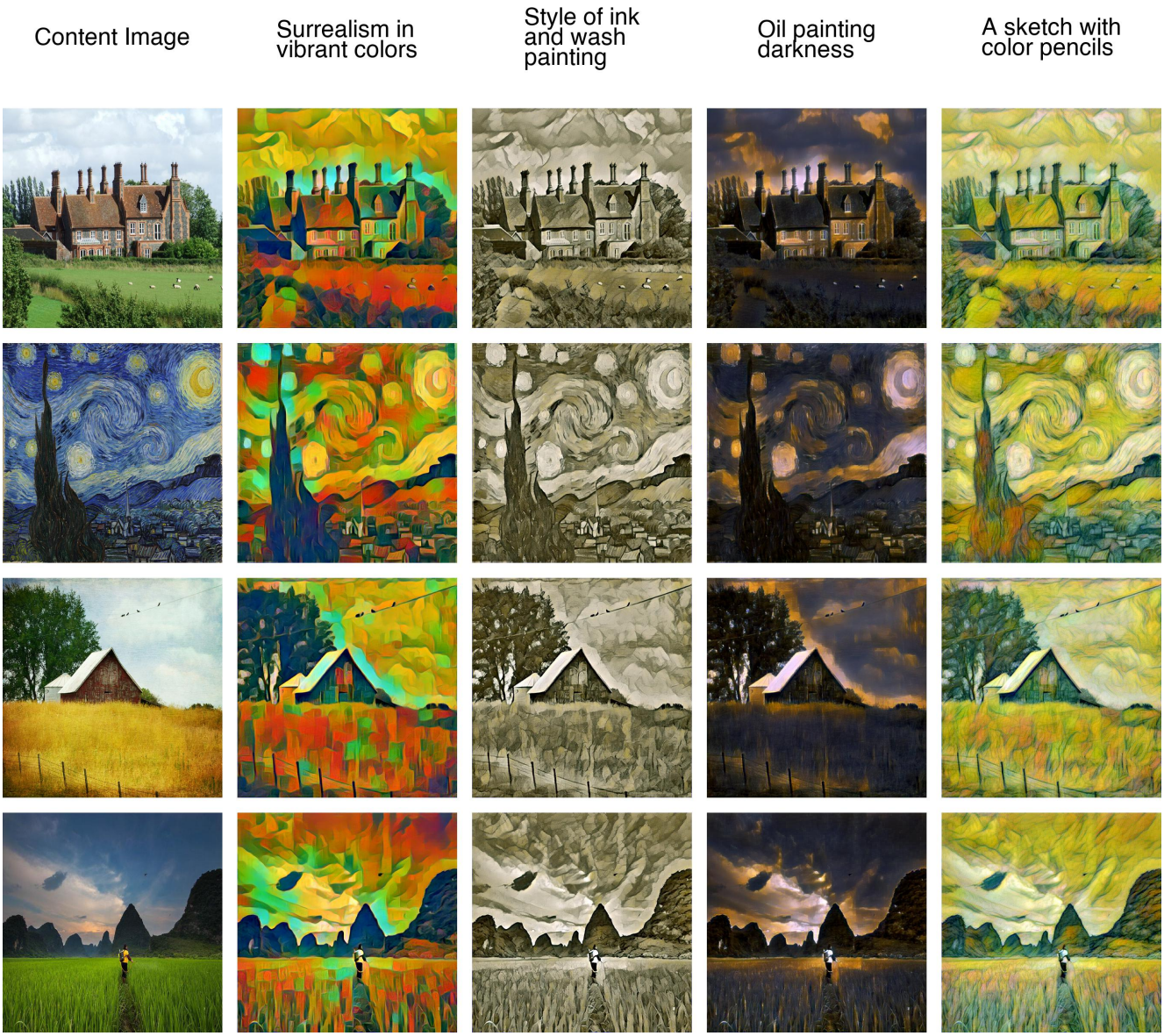


Figure 4. Performance of our FastCLIPstyler model on art style prompts.

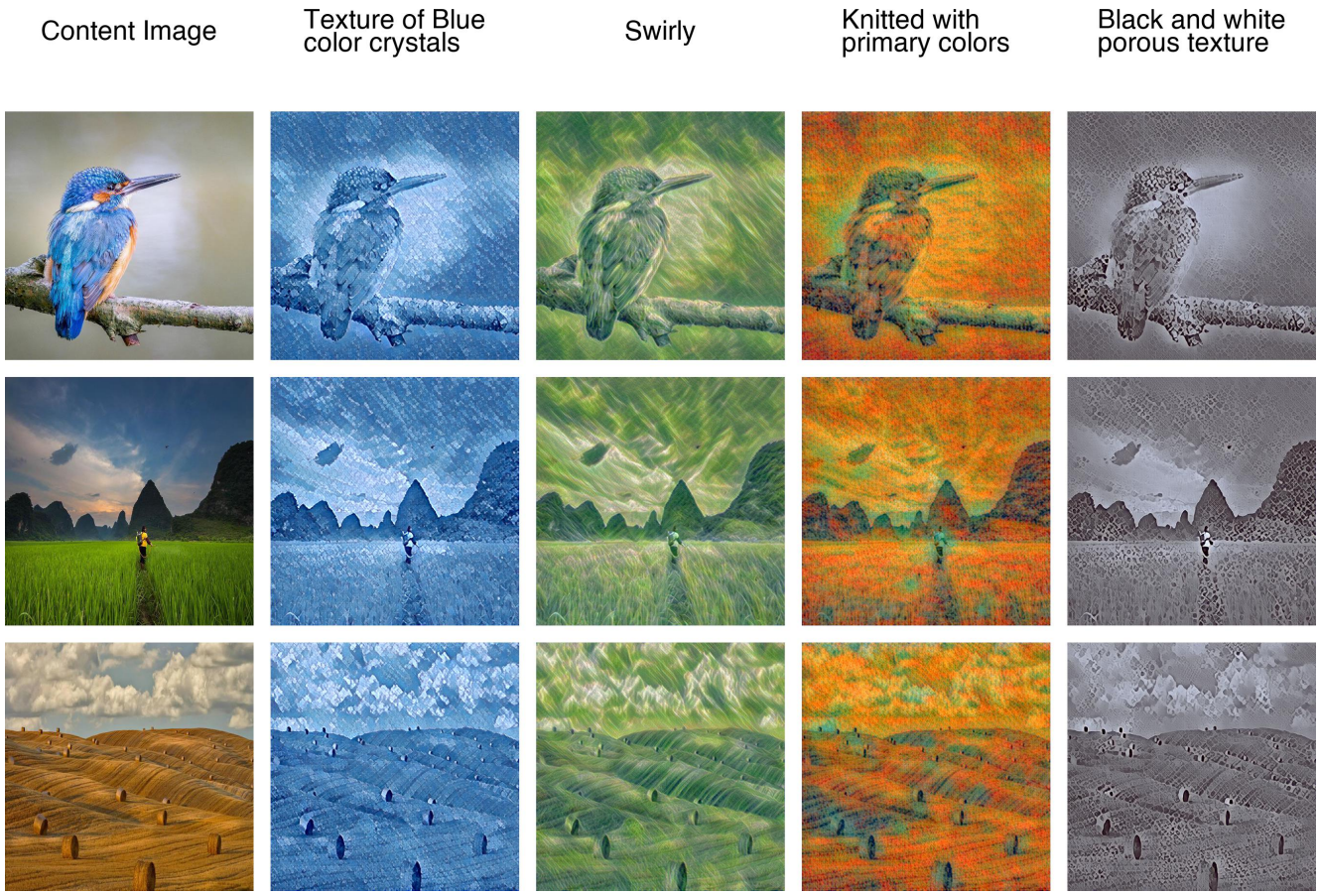


Figure 5. Performance of our FastCLIPstyler model on texture style prompts.

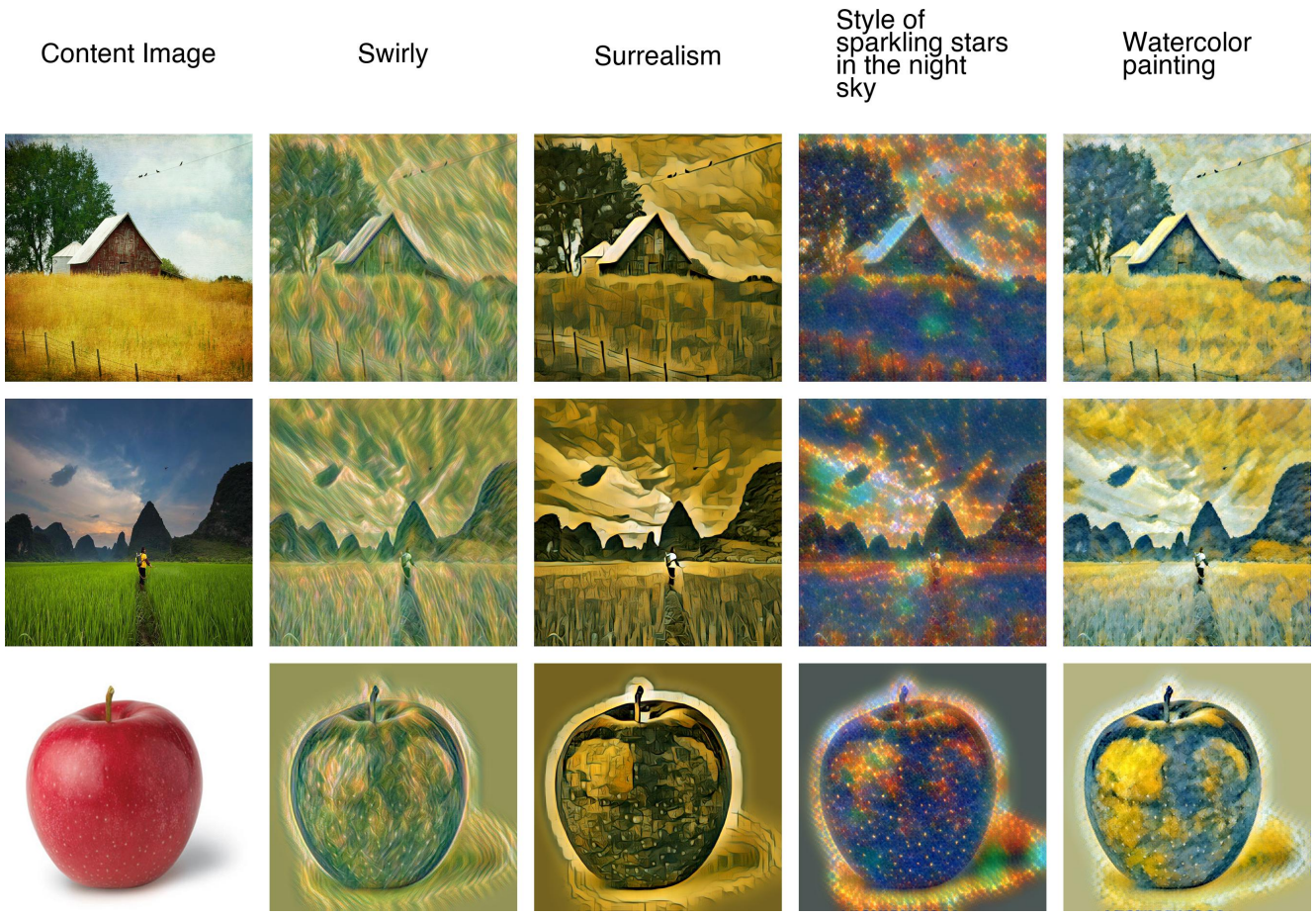


Figure 6. Performance of our EdgeCLIPstyler model on various prompts.

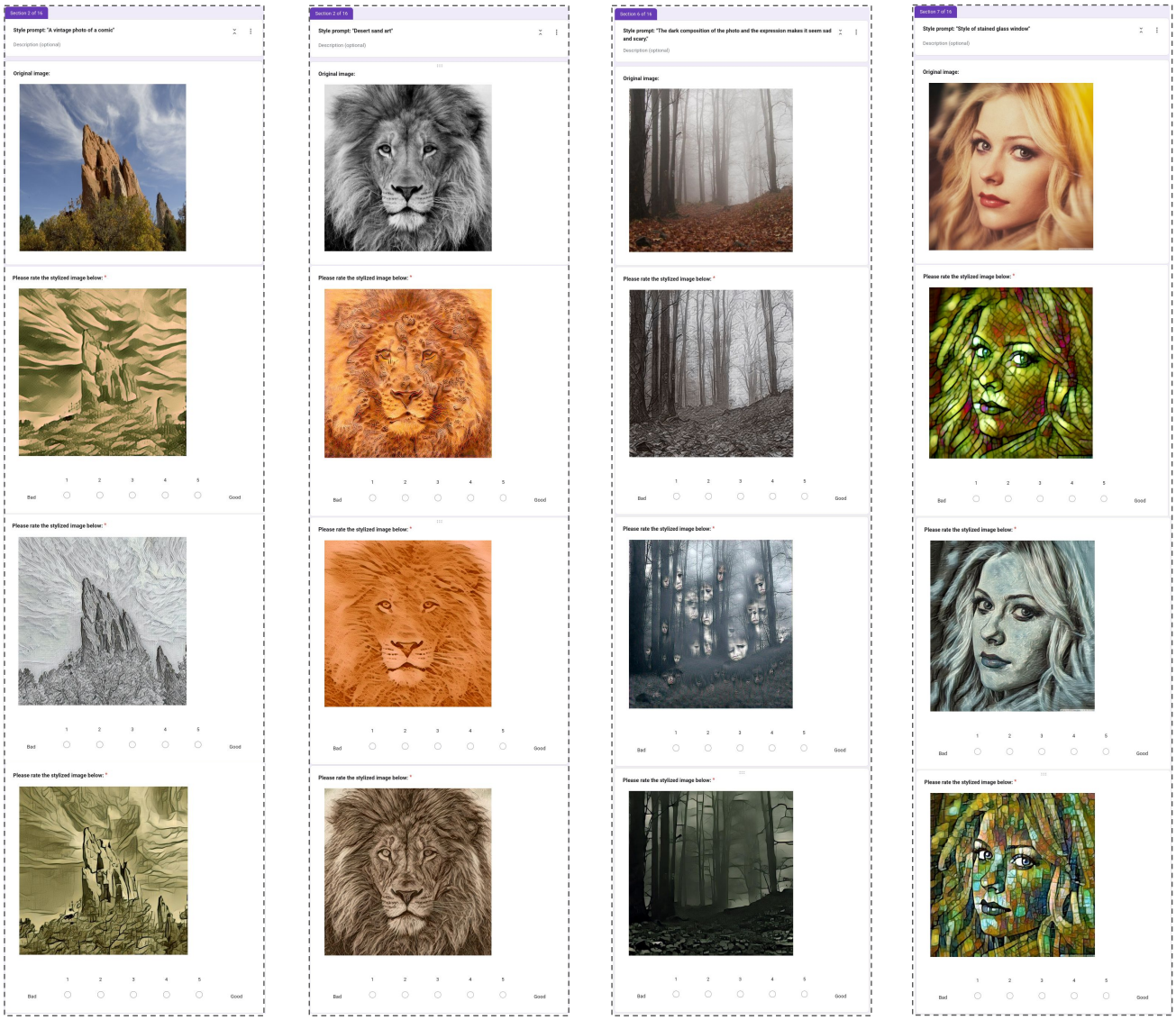


Figure 7. Sample screenshots of the forms sent to the participants.

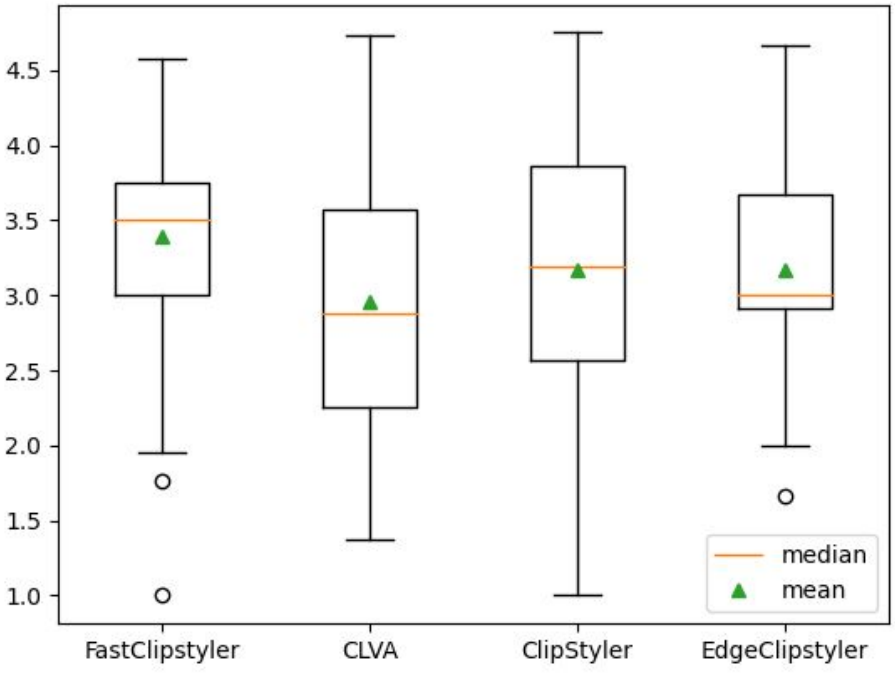


Figure 8. Result from user study as box plot

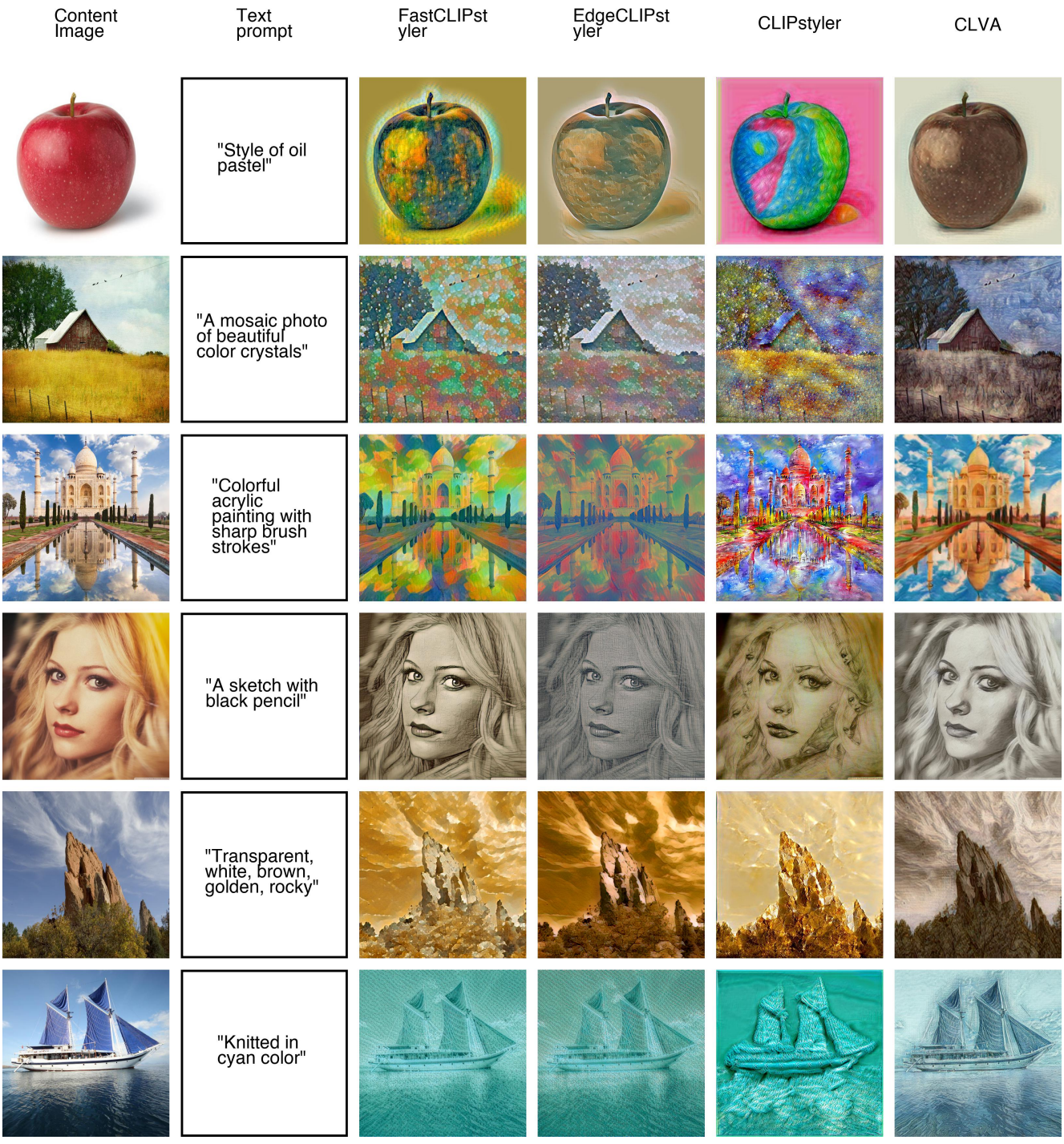


Figure 9. Comparison of our models with CLIPstyler and CLVA.