Report on Semantic Matching Research

September 14, 2024

Student : Yu Ting CHENG

Supervisor : Prof. Kai HAN

The University of Hong Kong

Contents

Abstract

Establishing point-wise semantic matches with semantic descriptors (i.e., vectors) among images is one of the fundamental challenges in computer vision. Some notable applications of semantic matching include object tracking, 3D reconstruction from 2D images, and image editing. To perform semantic matching, the semantic meaning of different parts of an image must be calculated. In order to find the semantic meaning within an image, a feature extractor will be used, where an input image was given to the extractor, and a feature map of the image will be generated. A robust feature extractor can create feature maps that can accurately represent the semantic meaning within the image. Recently, there have been novel methods in extracting feature maps from images. Such methods extract information from the latent layers of UNet structure in image generating models such as Stable Diffusion. These methods were based on the idea that some latent layers within the UNet structure should have semantic understanding of the image in order to calculate the output images. However, such methods have only used one latent layer as the sole feature map in semantic matching, and may not have fully utilised the potential of other latent layers within the Stable Diffusion pipeline. In this report, we will discuss the use of extracting multiple latent layers of different timestep, layers as well as learnt prompt embeddings, in order to extract feature maps that aims to improve accuracy on a higher resolution.

1 Introduction and Related Work

1.1 Feature maps and feature extractors

Semantic descriptors, or features, are vectors that convey semantic meaning and definition. These vectors have dimensions as high as hundreds of entries. By calculating the cosine similarity or dot product between features, we can see whether those features are similar. Feature maps are similar to normal images, but consist of features instead of the original Red, Green, and Blue values within a pixel. Given an image, a feature extractor can calculate and produce a feature map corresponding to the image. If the features found in two different images are similar, the corresponding locations of the features should would carry similar semantic meaning. A robust feature extractor can calculate feature maps with

features that could accurately represent the semantic meaning of the contents in the image. By finding a robust feature extractor, point-wise semantic matches can be established with higher accuracy. Our goal is therefore to find a robust feature extractor by using the latent layers within the Stable Diffusion pipeline.

1.2 Previous Work on Stable Diffusion and feature extraction

Recent works [5, 6, 15, 16]have used the latent layers in Stable Diffusion as means to extract feature maps. It was proposed that since image generating models such as Stable Diffusion could perform well on image-to-image translation [7, 12], such models will have some semantic understanding of the given image.

[16] introduced the use of pre-trained Stable Diffusion as feature extractors. This allows for a novel method of feature extraction to generate feature maps for various computer vision tasks.

SD+DINO[15] introduced the use of Stable Diffusion 1-5 for feature map extraction, and discovered the spatial awareness within the Stable Diffusion extracted feature map, and have fused the feature map with feature map from DINOv2 [9], another type of feature extractor. Since DINO features and Stable Diffusion features have different properties, they could be used to compliment each other to further improve matching accuracy.

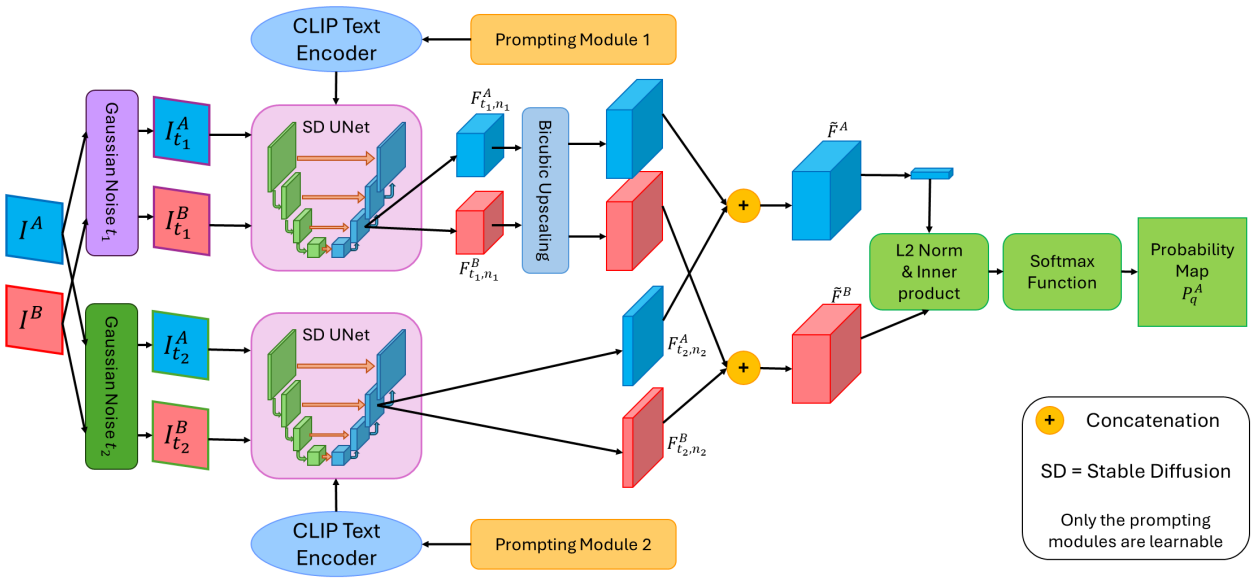
DIFT[6] uses a newer model, Stable Diffusion 2-1 as a standalone model for feature extraction. They have also performed a gridsearch in order to find the optimal timestep for feature extraction. The method has achieved similar performance as [15].

SD4Match[5] suggested learning the prompt embeddings passed to the stable diffusion model in order to improve on the results. The method also follows the principle of DIFT, but includes training on the prompt embedding. By learning the suitable prompts to condition on the diffusion model, this method has significantly improved the performance of matching.

2 Method

In this section, we will introduce our method, where we concatenate two feature maps extracted from two different latent layers of the Stable Diffusion pipeline, and also explore the possibility of using prompt tuning to enhance the performance. Section 2.1 will discuss preliminary information on Stable Diffusion, the use of the UNet structure to generate feature maps and the general pipeline of our method. Section 2.2 will discuss the use of upsampling in improving the accuracy of the feature maps. The dual Stable Diffusion Structure will be further discussed in Section 2.3. Section 2.4 will then discuss the use of SD4Match[5] prompt tuning on the dual Stable Diffusion method.

Figure 1: Illustration of the general pipeline of our method



2.1 Preliminary

Stable Diffusion is an image generating model utilising the image diffusion model proposed in [4, 11]. The process includes a forward and reverse process. In the forward process, Gaussian noise is continuously added to a clean image I to corrupt the image. The corruption can be represented by the following equation:

$$I_t = \sqrt{\alpha_t} I_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

In the equation, ϵ follows a normal distribution, and α_t is a coefficient related to the level of corruption at timestep t. As t increases, more noise will be added to the image, and the image will be less recognisable When t reaches a value sufficiently large, T, image I_t is totally corrupted and resembles Gaussian noise. The diffusion model, $f_\theta(I_t, t)$, accepts the corrupted image I_t and timestep t, and will learn to predict the noise added at timestep t, which is ϵ_t. By removing the noise ϵ_t, the learnt diffusion model can therefore recover the original image, I_0. Various types of input can be conditioned to the reverse process to adjust the image output. To allow for easier interpretation, we will directly refer to the UNet's input as image I instead of the VAE encoded representation in later parts.

Within the decoder of the UNet structure in Stable Diffusion models, it was found out some intermediate layers carry semantic meaning and even spatial information [15] of the image, and therefore could be utilised as feature maps. Moreover, observations show that with a larger timestep and a earlier layer of the UNet structure, the feature map produced captures more abstract and semantic information, while local details and textures can be found in feature maps extracted from a smaller timestep and at a later layer. Our method will attempt to utilise the different characteristics of the two types of feature maps mentioned above in attempt to achieve results with the best of both worlds. Below is an explanation of the mechanism of our method.

For an image pair with two images I_t^A and I_t^B with dimensions of $\mathbb{R}^{3\times H\times W}$, which are both corrupted to a timestep of t, the UNet model $f_n(\cdot)$ in Stable Diffusion extracts their corresponding feature maps from the nth latent layer to produce feature maps $F_{t,n}^A$ and $F_{t,n}^B$ with dimensions of $\mathbb{R}^{C\times h_n\times w_n}$.

$$F_{t,n}^A = f_n(I_t^A, t, \theta)$$

In the above, C represents the number of feature channels, $H \times W$ represents the dimensions of the original image, $h_n \times w_n$ represents the dimensions of the feature map extracted at latent layer n of the UNet, $\theta \in \mathbb{R}^{N\times D}$ represents the the prompt embedding to condition the Stable Diffusion model. N is the prompt length while D is the dimension of the embedding.

For our method, we first extract two latent layers of the UNet model and concatenate them together along the feature dimension to form a larger feature map $\tilde{F}^A$. The reasoning will be further discussed in sections 2.2 and 2.3. Since the feature maps extracted from different latent layers may have different resolutions, we first bicubic upsample the feature map with lower resolution, so that both feature maps share the same resolution. The final feature map produced from our process, $\tilde{F}^A$, can be mathematically represented as

$$\tilde{F}^A = \text{bicubic}_{(h_{n_2}, w_{n_2})}(F_{t_1,n_1}^A) + F_{t_2,n_2}^A$$

In the formula, $\text{bicubic}_{(h_{n_2}, w_{n_2})}$ represents bicubic upsampling to a dimension of $h_{n_2} \times w_{n_2}$ (i.e. the height and width of the feature map extracted at the n_2th latent layer). The + sign represents the concatenation along the feature channel. When the final feature maps $\tilde{F}^A, \tilde{F}^B$ is calculated, we follow evaluation methods similar to that of SD4Match[5]. The feature maps, $\tilde{F}^A$ and $\tilde{F}^B$ are L2-normalised along the feature channel to form $\hat{F}^A$ and $\hat{F}^B$. Ground-truth correspondence information of image pairs (I^A, I^B) is represented as $\boldsymbol{W} = \{(\boldsymbol{w}_q^A, \boldsymbol{w}_q^B) | q = 1, 2, \ldots, n\}$. For each query point $\boldsymbol{w}_q^A = (x_q^A, y_q^A)$ in I^A, we can find its corresponding feature $\hat{F}_q^A \in \mathbb{R}^C$ within the feature map $\tilde{F}^A$and compute a correlation map $M_q^A \in [-1, 1]^{h\times w}$, with the feature map of B, $\hat{F}^B$.

$$M_{q,kl}^A = (\hat{F}_q^A)^\intercal \hat{F}_{kl}^B$$

$h \times w$ represents the resolution of $\hat{F}^B$, and $\hat{F}_{kl}^B$ represents the entry at (k, l) in $\hat{F}^B$. (i.e. $M_{q,kl}^A$ shows a correlation score between every location in the target image and the query point). A probability map P_q^A is then calculated by using the correlation map and softmax function $\sigma(\cdot)$ with temperature β:

$$P_{q,kl}^A = \sigma(M_{q,kl}^A) = \frac{\exp(M_{q,kl}^A/\beta)}{\sum_{ab}\exp(M_{q,ab}^A/\beta)}$$

The equation shows that a probability of a match between the query point and every location in the target image was calculated and is stored in the probability map P_q^A. With the probability map calculated, we can find the predicted matching point in image I^B.

2.2 Upsampling

For most feature extractors, the dimensions of the feature map are usually smaller than that of the image. This is because extraction of features and/or semantic information will require the information of a general neighbourhood of pixels. In DIFT[6] and SD4Match[5], the feature map extracted has its length and width that are both 1/16 of the original input image. Due to the reduction in resolution, semantic matches may only be able to match on general areas instead of focused points. It was found out that by upsampling the dimensions 2× using bicubic method could slightly improve the results when the margin of error is smaller. The reason might be that bicubic upsampling can take multiple pixels into account, and therefore provide upsampling that is similar to how the image was originally expressed, if upsampled to a low magnitude.

However, it is probable that using primitive upsampling methods on the feature map without any reference to the original image will not truthfully capture the details in the original image, especially when upsampling to a higher magnitude.

Dataset	SPair-71k[8]		PFPascal[2]		PFWillow[3]	
Margin of error	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.1$
DIFT[6]	39.34	52.31	70.25	84.43	46.31	72.22
2× Bicubic upsampled DIFT	40.93	52.61	70.12	84.20	48.08	72.58

Table 1: Comparing the performance of the feature maps by DIFT and 2× bicubic upsampled DIFT on SPair-71k, PF-Pascal and PF-Wilow datasets. We can see that bicubic upsampling may improve accuracy slightly when the margin of error is smaller ($\alpha = 0.05$), but have insignificant change when the margin of error is larger ($\alpha = 0.1$)

2.3 Dual Stable Diffusion Feature

As suggested in [15], different latent layers in Stable Diffusion captures different semantics and details of the image. While abstract semantic information can be found in earlier layers of the decoder with a larger timestep, local details and texture with higher resolution can be found in later layers with a smaller timestep. In order to differentiate between the local details among the semantically close general areas, we suggest to also utilise later layers of the decoder in order to capture more accurate matches within a smaller region. Note that in DIFT[6] and SD4Match[5], only the second latent layer has been extracted as a feature map since it can provide the highest matching accuracy. From searching different layers of decoder in Stable Diffusion 2-1, it seems most plausible to use the third layer of the decoder as a compliment to the second layer for our method. We then performed a gridsearch and found out timestep $t = 350$ has the highest matching score when the third layer was tested alone. Since general semantic information have a higher significance than local details on semantics, our method uses a weighting of 0.8:0.6 for the features in the second layer and third layer respectively. Due to the different resolution of the latent layers extracted, the feature map from layer 1 was upsampled bicubicly from testing with the prompt of "A photo of a {class}". Despite the timestep of t=350 scoring higher when the third latent layer was tested alone, it was found out that a timestep of $t = 200$ for the third layer has a higher score for matching when the second layer has a timestep of $t = 261$ and the two layers are concatenated together. In other words, we use $t_1 = 261, n_1 = 2, t_2 = 200,$ and $n_2 = 3$ for the results in Table 2.

We can see that from Table 2 that without any training, the inclusion of the third layer have performance only similar to or slightly better than when the original layer was upsampled. However, with some modifications, the use of multiple layers may have greater improvements.

Dataset	SPair-71k[8]		PFPascal[2]		PFWillow[3]	
Margin of error	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.1$
2× Bicubic upsampled DIFT[6]	40.93	52.61	70.12	84.20	48.08	72.58
DIFT with dual layers	41.21	52.52	70.75	85.14	48.17	72.34

Table 2: Comparing the feature maps with and without the inclusion of the third latent layer, there is only limited increase in the performance even when the magrin of error is small ($\alpha = 0.05$). However, this may be due to both latent layers sharing the same prompt and therefore only extracting similar properties and features.

2.4 SD4Match[5] prompt tuning

As mentioned in [5], by learning specific prompt embeddings for the Stable Diffusion pipeline, there could be great improvement in matching accuracy. Therefore, we have also implemented prompt tuning in our method. Since the two latent layers are capturing different properties of feature in the image, it would be ideal to have separate prompt

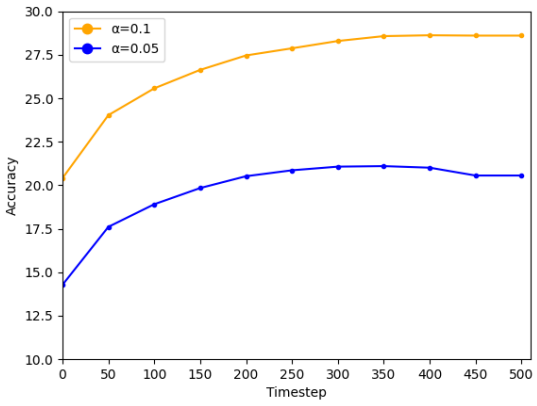


Figure 2: Gridsearch of performance in Layer 3 in semantic matching alone

embeddings for the two different latent layers to allow for more variation in the feature maps generated by the two latent layers. Note that the two prompt embeddings are trained together.

By using probability map P_q^A, the loss function can be calculated for training. The loss function $\mathcal{L}$ is calculated as the following:

$$\mathcal{L} = \frac{1}{n}\sum_{q=1}^{n} H(P_q^{A,gt}, P_q^A)$$

, where $P_q^{A,gt}$ is the Dirac delta distribution $\delta(\boldsymbol{w}_q^B)$ (i.e. the value at all locations are 0, except for $\boldsymbol{w}_q^B$ where the value is 1). A $k \times k$ Gaussian kernel is applied to $P_q^{A,gt}$ for label smoothing. Only the prompt embedding inputs will be trained, while the Stable Diffusion model will not have any changes. When only a universal prompt embedding θ is used for all images, this option is referred as **Single**. When a prompt embedding set $\Theta = \{\theta_1, \theta_2, \theta_3, \ldots, \theta_n\}$ is used, where each θ_k refers to the prompt used for a specific class, this option is referred as **Class**. When a prompt embedding is generated based on the two images with the use of a featuriser, DINOv2 in the case of our method as well as SD4-Match, to generate a unique prompt embedding, this option is referred as **Conditional Prompting Module**, or **CPM** for short. Note that in our method, there are two sets of prompt embedding modules for each feature map extracted.

Similar to [5], we can see that the timestep with the highest accuracy after training may not be the same as that before traing. For our case, the optimal timestep for the third layer is t=150, which differs from the original training timestep of t=350. Detailed results can be found on the table in section 3.3

3 Implementation and Results

3.1 Implementation Details

Our method uses PyTorch[10] and Huggingface[1, 13, 14] repositories on Python. The Stable Diffusion model used was of version 2-1 with four latent layers. The model was trained on two V100 GPUs. For prompt tuning in SD4Match [5], using the SPair-71k dataset [8], Single and Class were trained for 5 epochs, while CPM was trained for 2 epochs. For prompt tuning in SD4Match, using the smaller PF-Pascal dataset [2], Single was trained for 50 epochs. The implementation details are the similar to that in SD4Match [5]. The layers and timestep selected for training are $t_1 = 261, n_1 = 2, t_2 = 350$, and $n_2 = 3$. We choose DINOv2-ViT-B/14 as the feature extractor in SD4Match-CPM. For Stable Diffusion, the maximum prompt length is 77, including two special tokens, which are SOS and EOS. Therefore, in SD4Match-Single and SD4Match-Class, we set the prompt length N = 75, and in SD4Match-CPM, we set N_{global} to 25 and N_{cond} to 50, occupying all 77 token positions. During inference, we set the timestep and layers as inference are set as $t_1 = 50, n_1 = 2, t_2 = 200$, and $n_2 = 3$. The temperature β and kernel size k is set to 0.04 and 7 respectively during training. The Adam optimizer was used for training, with a batch size of 9 for $30,000$ steps across all experiments. The learning rate is set to 1×10^{-2} for all configurations, except for the two linear layers $g_d(\cdot)$ and $g_n(\cdot)$ in CPM, where it's set to 1×10^{-3}. This learning rate remains constant throughout the training process. Finally, we resize all images to 768×768 for both the training and testing phases.

3.2 Dataset and Evaluation metrics

Our method was tested and evaluated with SPair-71k dataset [8] and well as PF-Pascal [2] and PF-Willow dataset [3].

PF-Pascal consists of 20 categories of objects, and contains 2941 training image pairs, 308 validation pairs, and 299 testing pairs. PF-Willow is the supplement to PF-Pascal with 900 testing pairs only. SPair-71k is a larger and more challenging dataset with 18 categories of objects with large scale and appearance variation between images. It contains 53,340 training pairs, 5,384 validation pairs, and 12,234 testing pairs. All three datasets has non-uniform numbers of ground-truth correspondences.

The common practice in literature is to use the Percentage of Corrected Keypoints (PCK) as the evaluation metric, which we followed in our method. In the evaluation metric, given an image pair (I^A, I^B) and their point-wise correspondence set $\boldsymbol{X} = \{(\boldsymbol{x}_q^A, \boldsymbol{x}_q^B)|q = 1, 2, \ldots, n\}$, for each $\boldsymbol{x}_q^A = (x_q^A, y_q^A)$, we find its predicted correspondence $\bar{\boldsymbol{x}}_q^B$. Note that $\boldsymbol{x}_q^B$ represents the point in image I^B in the qth correspondence in the image pair, and $\bar{\boldsymbol{x}}_q^B$ represents the predicted point in image I^B from its corresponding point $\boldsymbol{x}_q^A$, predicted by the trained model . We then can calculate the PCK for the image by the below formula:

$$PCK(I^A, I^B) = \frac{1}{n}\sum_{q}^{n} \mathbb{I}(||\bar{\boldsymbol{x}}_q^B - \boldsymbol{x}_q^B|| \leq \alpha * \theta)$$

In the equation, θ represents the base threshold and varies depending on datasets, while α is a number between 0 and 1 which represents the maximum margin of error, and $\mathbb{I}(\cdot)$ is the binary indicator function that returns 1 for true and 0 for false. The equation therefore calculates the number. For the PF-Pascal dataset, the base threshold θ is set as $\theta_{img} = \max(h_{img}, w_{img})$, which is the highest value between the height or the width of the image. For the PF-Willow dataset, the base threshold θ is set as $\theta_{kps} = \max(\max_q(x_q^B) - \min_q(x_q^B), \max_q(y_q^B) - \min_q(y_q^B))$, which is the maximum difference in height or width between two keypoints. For the SPair-71K dataset, the base threshold θ is set as $\theta_{bbox} = max(h_{bbox}, w_{bbox})$, which means the highest value between the height and width of the bounding box. The choices of these base thresholds align with the standard practices and conventions in the literature for these datasets.

3.3 Results

From Table 3, we can see that for the more challenging SPair-71k dataset[8], the inclusion of the third latent layer along and proper adjustments of prompt embedding could allow for significant accuracy improvement when the margin of error is smaller, since our method has out-performed the second-best method, SD4Match[5], by 8 percent when $\alpha = 0.05$. However, from Table 4, when the margin of error have increased to $\alpha = 0.1$, our method only outperforms the second-best method by 2 percent. This indicates that the inclusion of the third latent layer focuses more on local details and is most effective when the margin for error is smaller. Furthermore, in Table 5, we can see that on datasets such as PF-Pascal and PF-Willow, where the data has less variation in scale and appearance, the inclusion of the third latent layer may not have significant change on the performance of the result. This indicates that for images with little variation, inclusion of the third latent layer may not be necessary.

Table 3: Class-wise Evaluation of SPair-71k Dataset with $\alpha = 0.05$ the best results in each class is **bolded**. Our method have the highest performance among all 18 classes of objects, and overall outperforms SD4Match[5] by 8 percent when the margin of error is smaller.

Method	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dog	Horse	Motor	Person	Plant	Sheep	Train	TV	All
DIFT	45.78	37.01	59.23	17.80	29.03	31.40	26.86	70.86	24.44	52.44	37.90	41.87	33.09	31.46	31.11	29.30	43.78	32.77	37.26
DIFT[6] bicubicly upsampled	46.67	40.04	63.09	18.44	29.86	32.42	27.62	72.31	26.99	54.80	39.75	42.26	34.30	33.51	34.89	29.53	46.17	35.85	39.09
DIFT[6] with dual layer	48.96	42.00	61.53	21.68	31.87	35.32	28.68	69.92	30.45	55.55	40.31	41.83	36.25	32.50	38.51	29.31	50.76	47.84	41.14
SD4Match-Single[5]	61.04	47.34	62.65	41.72	42.69	61.55	57.47	71.96	48.91	64.36	51.58	55.44	46.88	56.64	44.13	36.64	65.58	56.41	53.64
SD4Match-Class[5]	65.23	49.19	67.56	47.25	43.13	73.87	58.76	75.14	49.22	61.10	52.09	55.48	54.65	66.72	40.96	40.19	72.80	68.14	57.40
SD4Match-CPM[5]	64.27	48.41	63.94	44.03	45.37	71.79	58.68	74.57	49.95	64.85	53.36	57.26	52.47	57.38	42.38	44.68	70.69	60.50	56.48
Single (Ours)	65.82	54.59	68.64	48.70	42.10	76.43	69.96	76.99	61.55	72.34	60.25	60.14	58.54	65.33	51.19	**48.57**	75.24	73.38	62.18
Class (Ours)	70.60	53.67	**76.34**	**53.00**	46.05	81.55	67.59	77.14	**65.53**	70.21	58.33	59.89	58.22	**75.16**	**55.62**	43.98	79.54	**77.27**	64.58
CPM (Ours)	**70.65**	**57.37**	74.16	51.14	**47.77**	**81.61**	**73.36**	**80.75**	62.71	**74.99**	**61.52**	**62.93**	**61.66**	64.66	51.71	47.30	**81.28**	74.42	**65.01**

4 Conclusion

In this report, we have explored the possibility of using multiple latent layers in Stable Diffusion for establishing point-wise semantic matches. We introduce our novel method, where two layers within the Stable Diffusion UNet structure was utilised in attempt to improve image semantic matching accuracy. We have shown that multiple latent layers can be utilised to allow for more accurate semantic matching with a smaller margin of error, and compared our results with similar methods and have improvements in accuracy when the margin of error is small.

Table 4: Class-wise Evaluation of SPair-71k Dataset with $\alpha = 0.1$ the best results in each class is **bolded**. Our method have the highest performance among 16 out of 18 classes of objects in the dataset.

Method	Aero	Bike	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dog	Horse	Motor	Person	Plant	Sheep	Train	TV	All
DIFT[6]	57.17	49.71	77.70	25.78	39.97	35.75	31.51	77.83	30.54	68.63	49.61	58.20	42.72	40.67	47.32	44.96	59.16	46.59	48.97
DIFT[6] bicubicly upsampled	57.18	49.76	78.39	25.36	40.28	36.28	31.71	77.85	31.42	68.93	50.24	58.09	42.85	41.54	48.48	45.44	59.73	47.66	49.39
DIFT[6] with dual layer	60.39	52.93	77.10	30.57	42.94	39.48	33.63	77.39	34.65	69.83	51.03	59.60	48.14	41.85	51.73	46.07	65.14	60.55	52.36
SD4Match-Single[5]	72.15	65.05	81.71	60.08	57.26	75.43	72.28	80.85	62.07	83.24	70.55	75.16	67.32	75.73	66.26	56.09	88.28	80.72	71.40
SD4Match-Class[5]	74.80	65.20	87.01	69.68	57.42	85.80	73.65	84.04	62.46	82.57	70.02	**76.77**	72.78	86.66	63.04	58.15	91.02	88.90	74.64
SD4Match-CPM[5]	74.74	66.14	85.04	63.26	**62.18**	85.80	75.49	82.65	64.32	85.68	71.77	77.74	70.13	77.19	64.62	63.17	90.99	84.54	74.43
Single (Ours)	72.90	65.46	85.60	63.87	55.98	83.22	80.69	82.94	68.78	85.07	73.11	73.72	68.33	80.53	68.54	**63.33**	88.94	89.72	74.64
Class (Ours)	75.91	64.28	**87.51**	**70.04**	56.62	**88.41**	76.48	82.14	**74.15**	85.28	71.17	74.24	71.88	**87.72**	**73.42**	59.92	90.65	**93.58**	76.58
CPM (Ours)	**76.80**	**67.48**	86.85	68.89	58.77	88.28	**81.91**	**84.99**	69.18	**85.97**	**74.68**	76.44	**72.92**	79.02	69.17	62.29	**92.35**	90.91	**76.67**

Table 5: Comparison with similar Stable Diffusion based semantic matching models on SPair-71k, PF-Pascal and PF-Willow datasets.

	SPair-71k $\theta_{bbox}@\alpha$		PF-Pascal $\theta_{img}@\alpha$		PF-Willow $\theta_{kps}@\alpha$	
	0.05	0.1	0.05	0.1	0.05	0.1
DIFT[6]	39.34	52.31	70.25	84.43	46.31	72.22
DIFT[6] bicubic upsampled	40.93	52.61	70.12	84.20	48.08	72.58
DIFT[6] with dual layer	41.21	52.52	70.75	85.14	48.17	72.34
SD4Match-Single[5](Trained on PF-Pascal)	29.29	43.84	79.59	92.44	48.28	76.97
SD4Match-Single[5](Trained on SPair-71k)	53.64	71.40	70.64	85.03	52.90	80.54
Single(Ours trained on PF-Pascal)	35.07	46.55	82.67	92.87	52.66	77.77
Single(Ours trained on SPair-71k)	62.18	74.64	71.74	85.83	54.61	78.09

Acknowledgements

This research project is made possible with the help of Prof. Kai Han for his continued guidance and suggestions, as well as Dr. Xinghui Li for his technical support. Moreover, I am grateful for this opportunity brought forth by the Laidlaw Undergraduate Research Scholarship and HKU Horizons Office. Thank you for providing me with this invaluable opportunity. Lastly, I would also like to thank my family and friends for providing emotional support during my research.

References

[1] Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate, 2022.

[2] Bumsub Ham, Minsu Cho, Cordelia Schmid, and Jean Ponce. Proposal flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3475–3484, 2016.

[3] Bumsub Ham, Minsu Cho, Cordelia Schmid, and Jean Ponce. Proposal flow: Semantic correspondences from object proposals. *IEEE transactions on pattern analysis and machine intelligence*, 40(7):1711–1725, 2017

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[5] Xinghui Li, Jingyi Lu, Kai Han, and Victor Prisacariu. Sd4match: Learning to prompt stable diffusion model for semantic matching. *arXiv preprint arXiv:2310.17569*, 2023.

[6] References Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[7] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2021.

[8] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Spair-71k: A large-scale benchmark for semantic correspondence. *arXiv preprint arXiv:1908.10543*, 2019.

[9] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[10] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[11] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[12] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. *arXiv preprint arXiv:2211.12572*, 2022.

[13] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

[14] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam ´ Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020.

[15] Junyi Zhang, Charles Herrmann, Junhwa Hur, Luisa Polania Cabrera, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. In *Thirtyseventh Conference on Neural Information Processing Systems*, 2023.

[16] Wenliang Zhao, Yongming Rao, Zuyan Liu, Benlin Liu, Jie Zhou, and Jiwen Lu. Unleashing text-to-image diffusion models for visual perception. *arXiv preprint arXiv:2303.02153*, 2023.

Appendix - Reflection on Summer Research Programme

Difficulties and Solutions

This research project has been quite a challenge as well as a great learning experience for me. The issues during the Laidlaw Scholar Research are the lack of background knowledge and the lack of time. Both of which contributes to most of my difficulties during my research. However, these challenges have made me learn so much in so little amount of time.

My background knowledge on Artificial Intelligence was meagre for me at the start of my research. Given that I was just a year one student, I have only learnt about some basic maths, probability theory, programming, computer architecture, and the use of Linux terminals. Most technical knowledge on AI will only be taught starting from year three. Throughout my research journey, I have faced various issues such as the software compatibility issues, hardware limitations, understanding code of job schedulers in servers as well as understanding the machine learning code. For most issues, I will first search for relevant information and potential solutions online, usually online forums will be sufficient to solve the issues. If I still cannot solve the issues, I will ask some PhD students or my supervisor for help.

It is without a doubt that I have learnt so much from the problems I have faced throughout this research journey, from technical skills like running code of AI models, to soft skills like learning to strike a balance between work and life.

Improvements to be made

If I can repeat this research process again, I will attempt to spend reasonably, explore a wider range of methods, and managing my time more efficiently during my research.

Not spending on paid computing services has cost me valuable time. I originally attempted to use Google Colab, a free cloud computing service provided by Google, to run some machine learning code. However, since there is no such thing as free lunch, Google Colab comes with many limitations. There are limited time of use of the GPU resources every day, your data get completely wiped if there were no activity for a certain amount of time, and you can only use at most one GPU for each process. These limitations on Google Colab has deemed it impossible for me to run the code. Looking back, it seems like my idea of saving money by using a free service has cost me about a month of precious time with little to no progress on my research.

Exploring alternative methods is also something that I have found to be important upon reflection on my research process. At the start of my research, I was hyperfixating on having the code to run successfully on Google Colab, and have neglected other possible platforms such as GPU server rental services and HKU High Performance Computing services. If I have explored the use of other computing services earlier, I could have made wiser choices on choosing between computing services, and could have saved much time on my research process.

Time management is another aspect that I could have improved on if I could repeat the research process. The research programme do not define clear boundaries between work and life for myself, and I have free reign to choose when to work and when to rest. There were times during my research process where I have overworked myself during the weekdays, and was in the verge of collapsing during the weekends. My routine basically consisted of sleeping, working, and eating only, which has made me quite tired throughout the day. It was until later where I have found balancing my work and rest and having a clear mind is more efficient than cramming every possible time into working and being unable to think straight most of the time. The number of hours spent on working do not always mean the amount of work done.

In conclusion, this research project has been a valuable learning experience, and the challenges I faced have taught me important lessons. By reflecting on my research endeavor, I hope I can better prepare for the upcoming Leadership in Action programme as well as other future opportunities.