# Learning to Prompt CLIP for Monocular Depth Estimation: Exploring the Limits of Human Language

Dylan Auty
Imperial College London
dylan.auty12@imperial.ac.uk

Krystian Mikolajczyk
Imperial College London
k.mikolajczyk@imperial.ac.uk

## Abstract

*CLIP is a significant vision-and-language training framework that has shown surprisingly general understanding of the world, with good performance in many open-ended tasks with little or no additional training. A recent technique has used CLIP to perform 0-shot Monocular Depth Estimation (MDE) by using depth-related prompts, but the use of human language in these prompts presents an unnecessary human bias. In this work, we use continuous learnable tokens in place of discrete human-language words to shed light on the problem. We achieve a significant boost in performance, and find that the learned tokens do not map neatly to depth-related human language, implying that CLIP's concept of depth is not succinctly expressible in human language. We posit that this may extend to other CLIP concepts, and believe that this finding will spark further research into both the use and interpretation of non-linguistic tokens in all open-ended scene interpretation tasks. Code is available at* `https://github.com/DylanAuty/PromptLearningCLIP-MDE`

## 1. Introduction

Monocular Depth Estimation (MDE) from a single image only is a poorly-posed problem: a given 2D image may have been taken from any number of 3D scenes. This inherent ambiguity makes the problem extremely challenging, requiring the use of scene context and a degree of prior knowledge to ascertain the most likely solution. Despite this, however, deep-learning methods are able to perform well on this task, although they are typically large and highly specialised.

Recently, the Contrastive Language-Image Pretraining (CLIP) [19] method has been proposed as a way to train a model jointly with both images their natural-language captions. This results in a model with significant and surprising general knowledge about the world, which has been used as the basis of several low- and zero-shot models for image classification, captioning, and visual question answering (VQA) (see section 2.2). The use of a very large dataset of captioned images enables the model to learn to associate *concepts* with *human language*, and it is this general association that makes CLIP useful for open-ended interpretation of the world. While this general knowledge is typically applied to tasks with open-ended linguistic output (e.g. classification, VQA, captioning etc.), it can be applied to tasks with scalar output spaces as well, such as depth estimation.

DepthCLIP [27] is a recent method that applies the general knowledge of pretrained CLIP models to MDE, phrasing it as a problem of zero-shot ordinal classification of image patches into depth-bins. Human-language prompts are created that each correspond to one of several bins of depth values, in the form 'This object is [near/far/close/distant/etc.]'. A pretrained CLIP model is used to embed the input image and each of the prompts, and the degree of correlation between the image features and each of the prompt features is used to determine the weight of that prompt's assigned depth bin. A weighted sum of the bin-centres produces the final continuous scalar depth output.

While DepthCLIP achieves impressive zero-shot performance, it suffers from several weaknesses. Human bias is introduced by the manual choice of words used for the prompts at each part of the ordinal depth scale, and the fact that the depth bin prompts are words at all. CLIP's training set is 400M captioned images, meaning that its understanding of abstract concepts that are not commonly discussed in captions - like depth - is likely to be implicit rather than explicitly expressible in words. Therefore, we posit that it is presumptuous to assume that human-chosen and human-language words would be the best way to convey an abstract concept like scalar depth to CLIP, and that human-designed captions are unlikely to be the most efficient solution.

A further human-language-related bias that DepthCLIP suffers from is the limitation on the number of depth bins that may be used. When using a conceptual description of depth as an ordinal scale (such as 'very near' or 'far' as opposed to directly stating numbers), any increase in scale granularity requires increasingly verbose and tortuous descriptive phrases. This naturally limits the number of different depth bins that can be used, in part due to a limitation of natural human language: the tokens (words) are inherently discrete.

**In this work, we introduce prompt learning for Monocular Depth Estimation using CLIP, and analyse the learned embeddings to obtain insight about how CLIP conceives of depth**. We build on the work of DepthCLIP by using **learnable prompt tokens** that are not affected by the human bias of being either chosen by a human or being made up of human language, delivering significant performance improvements over the baseline. With the training of only a few thousand token parameters combined with a pretrained and frozen CLIP, we produce a significant performance improvement across all metrics when compared to either random controls or to various human-language depth bin prompts. As the learned prompts are not input-dependent, our results may indicate that the learned tokens represent the *generic* concept of a particular depth range better than the human-language prompts used. We analyse the learned tokens to better understand why human language is insufficient for the task and how CLIP-like models understand the concept of depth. It is our hope that this analysis will extend to other abstract concepts and so can be leveraged for any system that uses a CLIP-style model as a way of interpreting textual or visual input, either by using better non-linguistic prompting or by encouraging additional research into the decoding of non-linguistic CLIP features into human-understandable output.

We perform ablations across the number of prompts used to describe the range of depths, the distribution of depth values assigned to each prompt, and the number of learnable tokens in each prompt. We evaluate our work on the KITTI[9] and NYUv2[23] datasets, for the outdoor and the indoor domains respectively.

Our main contributions are:

1. **We introduce learnable prompt tokens to prompt CLIP for Monocular Depth Estimation (MDE).** Our system builds on and improves the work of DepthCLIP [27] by removing human biases caused by the use of human language.

2. **We perform extensive experiments to find optimal prompting strategies and templates.** We experiment with different numbers of depth-bin prompts, the use of learnable context tokens, and different depth-bin distributions.

3. **We analyse and interpret the learned prompts,** providing insight into the nature of the CLIP latent space and the suitability of language for prompting large language models in general. Our work gives compelling evidence that **human language is inefficient in precisely explaining the concept of depth to CLIP**, with consequences for future works that seek to better exploit its understanding of an open set of concepts.

## 2. Related Work

### 2.1. Monocular Depth Estimation (MDE)

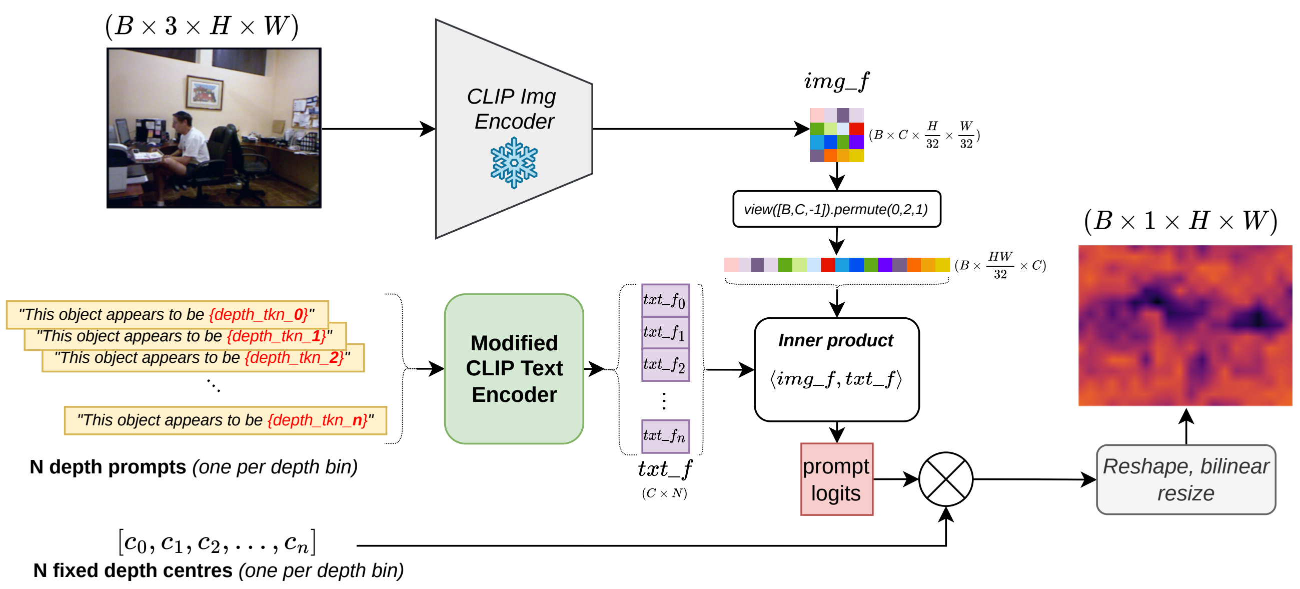
Monocular depth estimation (MDE) has been successfully tackled by deep-learning systems. Most approaches treat the problem as an image-to-image translation problem, using an encoder network to obtain high-dimensional, low-resolution image features, and then upsampling these features using a decoder network. Different methods make minor variations to this basic model structure to encode some additional inductive bias or to enforce specific losses. Multi-scale or coarse-to-fine approaches aim to improve consistency of neighbouring regions while still providing detail [13, 7, 6]. Multi-task approaches try to introduce information from other tasks that may have use for depth estimation, or try to make use of the information from other tasks (most commonly semantic segmentation)[20, 12, 2, 26, 25].

Recently, the MDE literature has shifted towards a new paradigm: rephrasing the problem as ordinal classification as opposed to pixelwise regression. Each pixel is assigned probabilities of belonging to one of several depth 'bins' distributed along the range of the dataset in question. A linear combination of the bin centres with the predicted probabilities provides a scalar depth value as output.

AdaBins [3] note that the bin distributions should be made adaptive rather than remaining fixed, and condition the distribution of the depth bins based on the input image using a transformer-based module. BinsFormer [15] and PixelFormer [1] improve on this idea with the addition of more complex transformer-based encoders and decoders. Most recently, LocalBins [4] applied the adaptive-binning technique on a per-pixel-region basis instead of a per-image basis, and is the state-of-the-art in the MDE literature at the time of writing. All In Tokens (AiT) [17] is a recent digression from the adaptive binning method: a VQ-VAE [18] autoencoder is trained on the ground-truth depth images, and a transformer-based encoder is trained to map the input image into the quantised feature space for use by the pretrained decoder, achieving excellent results.

### 2.2. Contrastive Language-Image Pretraining (CLIP)

Contrastive Language-Image Pretraining (CLIP) [19] is a recent training framework that jointly trains image and text encoders. The CLIP model comprises a visual encoder and a text encoder, and is trained on 400M images with corresponding English-language captions. The training objective is to minimise the cosine distance between the features



$(B \times 3 \times H \times W)$
CLIP Img Encoder
$img\_f$
$(B \times C \times \frac{H}{32} \times \frac{W}{32})$
view([B,C,-1]).permute(0,2,1)
$(B \times \frac{HW}{32} \times C)$
$(B \times 1 \times H \times W)$
"This object appears to be {depth_tkn_0}"
"This object appears to be {depth_tkn_1}"
"This object appears to be {depth_tkn_2}"
"This object appears to be {depth_tkn_n}"
N depth prompts (one per depth bin)
Modified CLIP Text Encoder
$txt\_f_0$
$txt\_f_1$
$txt\_f_2$
$txt\_f_n$
$txt\_f$
$(C \times N)$
Inner product
$\langle img\_f, txt\_f \rangle$
prompt logits
Reshape, bilinear resize
$[c_0, c_1, c_2, \ldots, c_n]$
N fixed depth centres (one per depth bin)

Figure 1. **An overview of the pipeline used**. The basic structure of the pipeline is the same as [27], but with our **modified CLIP encoder** used instead to allow the use of **learnable tokens in place of human words** in the prompts. The modified CLIP text encoder is detailed in figure 2. The pretrained CLIP model is completely frozen; the only parameters that we train are those in the learnable tokens. **Note that the output prediction is low-resolution by design:** our aim is to probe the limits of CLIP's understanding without the confounding factor of a specialised learned decoder.

of an image and the features of its corresponding caption, extracted using the visual and text encoders respectively.

The trained CLIP model exhibits significant general knowledge and understanding of the world, and the authors apply it to zero-shot classification. In this configuration, an image is encoded using the trained CLIP image encoder. A textual prompt in English is constructed for each possible class in the form 'This is a photo of a(n) {classname}.', and the trained CLIP text encoder is used to extract features for each of these. The class whose corresponding prompt is most similar to the input image in feature space is deemed to be the class of the image. This approach requires no additional training, achieving excellent zero-shot performance.

Other works have leveraged CLIP's zero-shot abilities for less obvious tasks. It has been applied with success to both image captioning and Visual Question Answering (VQA): [5] use a few small linear layers to convert the combined prompt (question) and image features into categorical VQA outputs, [24] do few-shot VQA by training an answer template generator that transforms the question into an answer-style statement that can be used with the standard CLIP zero-shot classification setup, while [16] use a simple mapping network to convert the image embeddings into prefix tokens to be fed to a frozen large language model (GPT2) for caption generation. While some of this performance may be attributed to the information likely to be in its training captions, some of these tasks may require a degree of reasoning (e.g. the ability to count or to accurately summarise information in an image).

The remarkable general knowledge exhibited by CLIP makes it a useful target of research for cross-domain, cross-task, and open-ended problems. Recently, DepthCLIP [27] extend the zero-shot classification approach to achieve good-quality (albeit necessarily low-resolution) zero-shot MDE performance. This good performance is particularly surprising considering that the commonplace image captions scraped from the internet that form CLIP's training dataset are not likely to contain precise, numerical scalar values of depth on a per-pixel basis. This indicates that CLIP has a more nuanced layer of understanding than would be admitted by the training captions, and/or that the 'meaning' extracted by its trained image encoder includes a degree of reasoning ability.

## 2.3. Prompt engineering

Modifying the prompts used for large language models (LLMs) have been shown to improve performance for certain tasks. [22] used a first-order approximation of gradient to select the optimal human-language words to use in prompting masked language models for sentiment analysis and natural language inference; although results were promising, the use of human language brings with it the biases detailed in section 1. The original CLIP paper[19] provide contextual clauses, e.g. "...a picture of a pet" is appended to the classification prompt to condition CLIP's responses for use with a pet-related dataset.

Instead of manually selecting the correct prompts to use, some works have experimented with learnable prompts. For VQA, [24] and [10] generate prompts based on the input questions and images to encourage LLMs to answer questions. A CLIP-base image captioning model [16] generates a prompt 'prefix' that is fed to a frozen generative LLM, which then generates a salient caption. CoOp [29] recognised that human words may not be the most suitable for providing context, and introduced learnable context tokens for use with classification. A successor work [28] uses the input image to condition the context tokens. In both cases, classification performance improves dramatically. VPT [11] introduces learnable tokens to a vision transformer setup, where the learned tokens are trained to improve performance on a specific task. None of these methods, however, directly encode specific input-agnostic concepts themselves in input tokens (as opposed to the general context for the problem or information about a specific input).

## 3. Method

In this section, we detail our method. We augment the method of DepthCLIP [27] by introducing **learnable prompt tokens** as a replacement for the human-selected, human-language words used to describe each of the depth classes, thus allowing the user to prompt the model for specific and indescribable concepts (such as depth). With the addition of only a few thousand trained parameters on top of the frozen pretrained CLIP model, our method produces significant performance improvements over human-language prompts. An overview of the pipeline is shown in figure 1; this pipeline matches that used in DepthCLIP with the exception of our modified text encoder, the detail of which is shown in figure 2.

### 3.1. Depth Bin Classification with Prompts

We follow the method of [27]: instead of classifying class labels, we perform depth bin classification. Mirroring the zero-shot classification strategies, we create one prompt per depth-bin, with the depth word selected being different between prompts to correspond to different bins.

The original CLIP language embedding pipeline first tokenizes the words in the input prompt, looking up frozen pre-trained 512-dimensional tokens for each in-vocabulary word. These tokens are then fed to the transformer-based language model to obtain the final text embeddings. Our method replaces certain human-language words within the prompts with special, out-of-vocabulary override words using our modified CLIP encoder (shown in figure 2). The corresponding 512-dimensional tokens for these override words are learnable parameters, allowing the learning of tokens for which no English word exists. This follows the technique shown in CoOp [29], who apply a similar technique to learn the context tokens surrounding the *class_name* token and obtain an increase in performance for classification (see section 2). Learned tokens are initialised by randomly drawing 512 elements from a normal distribution with mean 0 and standard deviation 0.02, and are optimised "in-place" as part of the depth estimation pipeline described. The objective is the minimisation of the loss function described in equation 1. Different datasets were used to train the tokens depending on the experiment (see section 4.1.2).

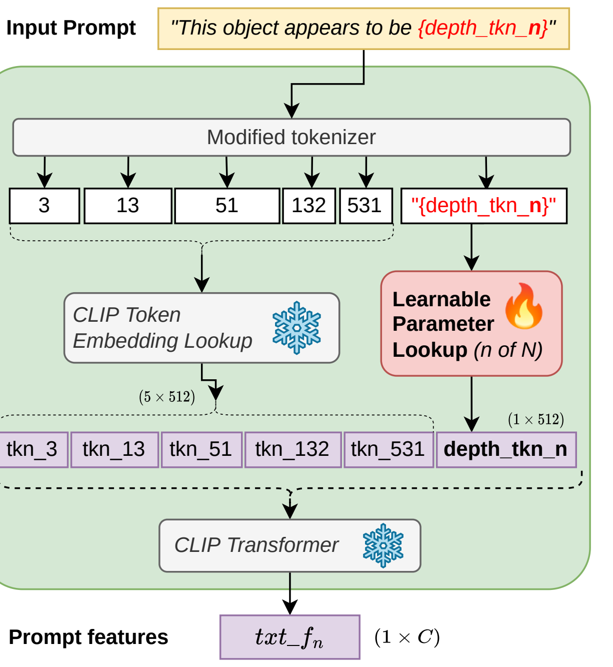


Figure 2. **Our modified CLIP text encoder**, used to allow the insertion of learnable tokens in place of the human-word tokens. Special words in the prompt are replaced by the tokenizer with learnable parameters (`nn.Parameter`). The pretrained CLIP token embeddings and transformer are frozen: the only parameters that are trained are the learnable tokens.

When comparing the use of learned tokens to the use of human language tokens, we use 7-point scales to maintain a balance between the expressiveness given by adding more bins and the difficulty in expressing a many-pointed ordinal scale with human language. However, learned tokens do not suffer from this limitation, and in section 4.4 we experiment with increased bin number using learned tokens instead of human-language words.

### 3.2. Scalar output

Each of the $N$ generated prompts is embedded using the CLIP transformer, producing text features of shape $[C \times N]$. The input image is encoded through a pretrained CLIP image encoder that has had the final pooling layer removed, providing patch-wise features of size $[C \times \frac{H}{32} \times \frac{W}{32}]$. $C$

| Name | Template |
|---|---|
| Baseline | *'This object appears to be* {depth_tkn}.' |
| 1o1d | '{p_0} *object* {p_1} {depth_tkn}.' |
| 1o2d | '{p_0} *object* {p_1} {p_2} {depth_tkn}.' |
| 4o4d | '{p_0} {p_1} {p_2} {p_3} *object* {p_4}{p_5} {p_6} {p_7} {depth_tkn}.' |

Table 1. **The prompt templates used.** One prompt is generated per depth-bin: the {depth_tkn} is replaced with either the learned depth token corresponding to that prompt's bin, or with the human-language words detailed in table 2, depending on the experiment. Context tokens ({p_i}) are learnable but identical between prompts. Template names describe the number and location of learnable context tokens: '$X$o$Y$d' denotes $X$ learnable context tokens before the word 'object', $Y$ more after it, and the {depth_tkn} at the end.

is the channel dimension, and is 1024 for the RN50 CLIP backbone used in our experiments.

The inner product of the image and text features is taken and then normalised to provide depth-bin logits of shape $[N \times \frac{H}{32} \times \frac{W}{32}]$. The vector of $N$ pre-determined depth-bin centres in metres, whose distribution is fixed at the start of the experiment, is then multiplied by these logits and summed along the $N$ dimension to give a low-resolution output depth map of shape $[1 \times \frac{H}{32} \times \frac{W}{32}]$. This is then bilinearly upsampled to the full input resolution to give the final prediction.

## 3.3. Prompt Templates

We experiment with different learned prompt templates, varying the number and location of the learnable tokens within the prompts. The templates we experiment with are shown in table 1. The most basic prompt template uses only a learnable depth token (one per depth-bin prompt). $N$ individual prompts will be generated according to the $N$ bins in use; between these, only the depth tokens will vary. The other learnable tokens in the prompt, if in use, remain the same between different bin prompts.

## 4. Experiments

In this section, we detail our experiments. We first describe our experimental setup. We then detail our experiments to confirm the efficacy of the prompt learning technique when applied to monocular depth estimation. We demonstrate the improvement over human-language prompts and random controls, then perform ablations to determine the optimal number of depth prompts, the optimal number of learnable tokens in each prompt, and the optimal distribution of depth bins across each dataset.

## 4.1. Experimental Setup

For repeatability, our experiments are all conducted with a batch size of 16. The random seed used is fixed for all experiments. All code is written in PyTorch. Training is performed for 25 epochs using either 2x 1080s, 1x A5000, or 1x A6000 NVIDIA GPUs. Evaluation is performed on the final training checkpoint only.

Our training hyperparameters mirror [3]: we use the AdamW optimizer with a learning rate of 0.000357 and a OneCycleLR learning rate scheduler, with max_lr=0.000357, cycle_momentum=True, base_momentum=0.85, max_momentum=0.95, div_factor=25, and final_div_factor=100.

The pretrained CLIP checkpoints we use are those provided by the official OpenAI CLIP repository[1]. We use the ResNet-50 backbone for all of our experiments, both for the sake of consistency and because several works have shown some evidence that the ResNet-based CLIP backbones are better able to extract spatially-relevant features than the Vision Transformer CLIP backbones [24, 21, 14].

### 4.1.1 Loss function

The loss function used is a variant of the Scale-Invariant Log-Loss (SILog), first proposed by [7] and modified by [3]:

$$\mathcal{L}_{SILog} = 10\sqrt{\frac{1}{K}\sum_{i\in K} g_i^2 + \frac{0.15}{K^2}(\sum_{i\in K} g_i)^2} \qquad (1)$$

where ground-truth and predicted depth values for pixel $i$ are given as $d_i^*$ and $d_i$ respectively, $g_i = log(d_i) - log(d_i^*)$ and $K$ is the total number of pixels with valid depth values. The learnable tokens are trained by minimising this loss function.

### 4.1.2 Datasets

We use the **NYUv2** dataset [23] to represent the indoor domain and the **KITTI** dataset [9] to represent the outdoor domain. The splits we use of these datasets are taken directly from the official implementation of [3][2], and our dataloader and data augmentation code is a slightly modified version of that used in the same repository.

**NYUv2** comprises a variety of indoor scenes with ground-truth depth captured using a Microsoft Kinect. We use 24231 training examples and the official test split of 654 examples. We train on random crops of size $[416 \times 544]$, and evaluate on the Eigen crop [7]. The input images at test time have dimensions $[480 \times 640]$.

**KITTI** is an outdoor driving dataset with sparse depth ground-truth captured using LiDAR. We use 23157 training and 696 test examples. Following [3], during training we take a random crop of size $[352 \times 704]$, and during evaluation we follow the cropping strategy of [8].

[1]github.com/openai/CLIP

[2]github.com/shariqfarooq123/AdaBins

| **Bin** | **depth-7** | **size-7** | **colour-7** |
|---|---|---|---|
| **0** | *very close* | *very small* | *very red* |
| **1** | *close* | *small* | *red* |
| **2** | *slightly close* | *slightly small* | *slightly red* |
| **3** | *neither close nor distant* | *neither small nor large* | *neither red nor green* |
| **4** | *slightly distant* | *slightly large* | *slightly green* |
| **5** | *distant* | *large* | *green* |
| **6** | *very distant* | *very large* | *very green* |

Table 2. **The 7-point human-language ordinal scales used as non-learned depth tokens**. We compare these to the use of learned tokens in section 4.2. The depth-7 and size-7 scales contain geometrically-relevant words, and the colour-7 scale is used to provide a control that is unrelated but still follows an approximate 'continuum' of meaning.

### 4.1.3 Evaluation Metrics

We use the commonly-used evaluation metrics defined in [7]: Abs relative difference (Abs): $\frac{1}{T}\sum_{i=1}^{T}\frac{|d_i-d_i^*|}{d_i^*}$, RMSE (RMS): $\sqrt{\frac{1}{T}\sum_{i=0}^{T}\|d_i-d_i^*\|^2}$, log RMSE (RMSL): $\sqrt{\frac{1}{T}\sum_{i=0}^{T}\|log(d_i)-log(d_i^*)\|^2}$, and the threshold accuracy $\delta_n$: % of $d_i$ s.t. $max(\frac{d_i}{d_i^*},\frac{d_i^*}{d_i}) = \delta < thr$, where $\delta_n$ denotes that $thr = 1.25^n$ (we use $n \in \{1,2,3\}$).

We follow [3, 13, 4] in computing metrics as a running average throughout evaluation, using a batch size of 1. We also use test-time augmentation: each input image is run through the model both as-is and mirrored in the y-axis. The prediction from the mirrored input is mirrored back, and the two predictions are averaged together pixelwise to give a final prediction on which all metrics are computed.

## 4.2. Comparing Learned to Human-Language Tokens

We first verify the effects of learning tokens as compared to the human-language tokens. The 'baseline' shown in table 1 is used, with the human-language or learned tokens replacing the {`depth_tkn`}. We use 7 depth bins, evenly distributed throughout the range of the target dataset: for NYUv2, this is [0.001, 10.0] metres, and for KITTI this is [0.001, 80.0] metres. Even bin distribution is used so as to remove bin distribution as a confounding factor, though recent work in adaptive bin distribution for depth shows that even bins are sub-optimal (see section 2.1). 7 depth bins are used to allow comparison between the use of learned depth tokens and human-language ordinal scales shown in table 2.

The human-language ordinal scales used are shown in table 2. The 'depth-7' scale relates directly to depth and the 'size-7' scale relates to it via the phenomenon of perspective. The 'colour-7' scale is included as a further control that does not relate to depth in any obvious way, but that still describes an approximate 'continuum' of meaning.

We also compare with random tokens (denoted as 'random-7') as an additional control: these are randomly initialised from a normal distribution with mean 0 and standard deviation 0.02 for each bin-prompt at the start of a run, then remain constant throughout the experiment. As an additional random control, we use randomly-selected natural language tokens from the CLIP token vocabulary (which includes word fragments). These are: *[summer, inian, fellow, greg, bis, ksh, aidan]* for the 7 bins.

| Depth tkns | Dset | Abs | RMS | RMSL | log10 | $\delta_1$ | $\delta_2$ | $\delta_3$ |
|---|---|---|---|---|---|---|---|---|
| colour-7 | nyu | 1.381 | 3.010 | 0.786 | 0.331 | 0.131 | 0.277 | 0.449 |
| size-7 | nyu | 2.130 | 4.431 | 1.281 | 0.446 | 0.048 | 0.119 | 0.235 |
| depth-7 | nyu | 1.014 | 2.413 | 0.566 | 0.265 | 0.227 | 0.431 | 0.606 |
| random-7 | nyu | 1.593 | 3.308 | 0.929 | 0.372 | 0.081 | 0.190 | 0.354 |
| rand-txt-7 | nyu | 1.335 | 2.875 | 0.754 | 0.324 | 0.129 | 0.287 | 0.471 |
| learned-7 | nyu | **0.319** | **0.970** | **0.139** | **0.128** | **0.465** | **0.776** | **0.922** |
| colour-7 | kitti | 2.177 | 23.470 | 1.328 | 0.446 | 0.077 | 0.163 | 0.267 |
| size-7 | kitti | 3.363 | 33.978 | 2.067 | 0.568 | 0.048 | 0.103 | 0.171 |
| depth-7 | kitti | 2.353 | 25.518 | 1.433 | 0.454 | 0.094 | 0.193 | 0.297 |
| random-7 | kitti | 1.664 | 19.279 | 0.994 | 0.370 | 0.119 | 0.251 | 0.400 |
| rand-txt-7 | kitti | 2.887 | 29.553 | 1.789 | 0.535 | 0.046 | 0.098 | 0.161 |
| learned-7 | kitti | **0.303** | **6.322** | **0.119** | **0.112** | **0.550** | **0.830** | **0.938** |

Table 3. **Comparison of human-word, random, and learned tokens** across a 7-bin scale, on the NYUv2 and KITTI datasets. Best results in bold, second best underlined. The use of only a single learned token in each prompt improves performance significantly across every metric. We also see that the geometrically-related human-language tokens are not always the best, particularly for KITTI where size-7 is outperformed by colour-7.

The results are shown in table 3. We find that the use of only a single learned token in each of the 7 bin prompts (a total of $512 \times 7 = 3584$ trained parameters only) significantly improves performance, by a factor of between $1.5$ to $5.5\times$. This is despite the even bin distribution and the low granularity resulting from only using 7 bins.

We also observe an interesting result in the case of the human-language tokens: depth-related words are not always the best. In the case of KITTI, we see that a random token performs better than any of the human-language word tokens. In the case of NYUv2, we see that the 'colour-7' tokens, chosen for their supposed lack of geometric meaning, outperform the size-related tokens. We conjecture that the most likely cause for these counterintuitive results is a general unsuitability of human language to describe the implicit and complex concept of depth, though it may also be due to either the distinctiveness of different random embeddings compared to a scale of depth-related words, or the distribution of the random embeddings being coincidentally ideal for mapping to the concept of depth.

## 4.3. Varying Number of Learned Tokens In Prompt

Having ascertained the learned depth tokens provide a significant performance boost, we turn our attention to the surrounding context of the prompt. Prior works have found that adding appropriate context to the prompt can improve performance; for instance, the original CLIP paper used the ending *"[...], a type of pet"* to improve classification performance on a pet-centric dataset [19]. CoOp [29] find

| Prompt format | Depth tkns | Dset | Abs | RMS | RMSL | log10 | $\delta_1$ | $\delta_2$ | $\delta_3$ |
|---|---|---|---|---|---|---|---|---|---|
| *Baseline* | depth-7 | nyu | 1.014 | 2.413 | 0.566 | 0.265 | 0.227 | 0.431 | 0.606 |
| 1o1d | depth-7 | nyu | $\underline{0.323}$ | 0.975 | 0.142 | $\underline{0.129}$ | 0.461 | 0.772 | $\underline{0.920}$ |
| 1o2d | depth-7 | nyu | $\underline{0.323}$ | $\underline{0.974}$ | $\underline{0.141}$ | $\underline{0.129}$ | $\underline{0.462}$ | $\underline{0.773}$ | $\underline{0.920}$ |
| 4o4d | depth-7 | nyu | **0.318** | **0.965** | **0.138** | **0.127** | **0.466** | **0.778** | **0.923** |
| *Baseline* | depth-7 | kitti | 2.353 | 25.518 | 1.433 | 0.454 | 0.094 | 0.193 | 0.297 |
| 1o1d | depth-7 | kitti | 0.331 | 6.528 | 0.132 | 0.120 | 0.511 | 0.809 | 0.929 |
| 1o2d | depth-7 | kitti | $\underline{0.321}$ | $\underline{6.420}$ | $\underline{0.127}$ | $\underline{0.117}$ | $\underline{0.527}$ | $\underline{0.817}$ | $\underline{0.932}$ |
| 4o4d | depth-7 | kitti | **0.309** | **6.334** | **0.122** | **0.114** | **0.541** | **0.826** | **0.936** |

Table 4. **Effect of adding learned context tokens {`p_i`} to human-language depth tokens** (using depth-7 ordinal scale). Prompt formats are described in table 1. Although {`p_i`} context tokens are the same for each depth-bin's prompt (unlike {`depth_tkn`}s), a significant performance improvement is still obtained.

that learning the context tokens, rather than manually setting them, provides a performance increase; we follow suit and experiment with the prompt types shown in table 1. 7 evenly-distributed depth bins are used, as in section 4.2.

In each of these prompt templates, the context tokens {`p_i`} remain constant between the $N$ different prompts. Only the {`depth_tkn`}s vary between each of the $N$ prompts. We first verify the efficacy of learnable context tokens {`p_i`} with human-language {`depth_tkn`}s in table 4, and then use learnable {`depth_tkn`}s and context tokens {`p_i`} together in table 5.

It can be seen that, for both datasets, learned context tokens produce a significant boost to performance. Note that context tokens {`p_i`} are identical for each depth bin's prompt, and therefore do not provide discriminative information to CLIP's language model. We also find that adding more learnable context tokens improves performance further, affirming that the findings of [29] apply for MDE as well as for classification.

It can also be seen that the combined use of both learned context tokens and learned depth tokens does not, in the case of KITTI, produce a significant performance over the use of only depth tokens. This may indicate that the existing template is sufficient to extract the most depth-relevant information from CLIP. For NYUv2, the use of learned context tokens improves only marginally over the use of only learned depth tokens, which may provide evidence to support this conclusion.

| Prompt format | Depth tkns | Dset | Abs | RMS | RMSL | log10 | $\delta_1$ | $\delta_2$ | $\delta_3$ |
|---|---|---|---|---|---|---|---|---|---|
| *Baseline* | learned-7 | nyu | 0.319 | 0.970 | 0.139 | 0.128 | 0.465 | 0.776 | 0.922 |
| ls4o4d | depth-7 | nyu | $\underline{0.318}$ | $\underline{0.965}$ | $\underline{0.138}$ | $\underline{0.127}$ | $\underline{0.466}$ | $\underline{0.778}$ | $\underline{0.923}$ |
| ls4o4d | learned-7 | nyu | **0.317** | **0.955** | **0.136** | **0.126** | **0.474** | **0.782** | **0.925** |
| *Baseline* | learned-7 | kitti | **0.303** | $\underline{6.322}$ | **0.119** | **0.112** | **0.550** | **0.830** | **0.938** |
| ls4o4d | depth-7 | kitti | 0.309 | 6.334 | $\underline{0.122}$ | 0.114 | 0.541 | $\underline{0.826}$ | $\underline{0.936}$ |
| ls4o4d | learned-7 | kitti | $\underline{0.307}$ | **6.209** | **0.119** | $\underline{0.113}$ | $\underline{0.546}$ | **0.830** | **0.938** |

Table 5. **Effect of combining both learned context and learned depth tokens**. Best results in bold, second best underlined. Some improvement from the combined use of both learned depth tokens and learned context tokens may be seen, but in the case of KITTI the results indicate that the majority of the performance may be attributed to the learnable depth tokens.

| Bins | Dist. | Dataset | Abs | SqRel | RMS | RMSL | log10 | $\delta_1$ | $\delta_2$ | $\delta_3$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 7 | Linear | nyu | 0.319 | 0.370 | **0.970** | **0.139** | **0.128** | **0.465** | **0.776** | **0.922** |
| 7 | Log | nyu | **0.312** | **0.366** | 0.991 | 0.142 | 0.130 | 0.449 | 0.767 | **0.922** |
| 20 | Linear | nyu | 0.314 | 0.360 | **0.949** | **0.134** | **0.125** | **0.475** | **0.785** | **0.926** |
| 20 | Log | nyu | **0.308** | **0.355** | 0.968 | 0.136 | 0.127 | 0.466 | 0.780 | **0.926** |
| 7 | Linear | kitti | 0.303 | 2.157 | **6.322** | 0.119 | 0.112 | 0.550 | 0.830 | 0.938 |
| 7 | Log | kitti | **0.265** | **1.926** | 6.933 | **0.112** | **0.108** | **0.580** | **0.838** | **0.939** |
| 20 | Linear | kitti | 0.292 | 2.062 | 6.159 | 0.113 | 0.108 | 0.573 | 0.841 | 0.941 |
| 20 | Log | kitti | **0.265** | **1.837** | **6.153** | **0.103** | **0.102** | **0.606** | **0.853** | **0.948** |

Table 6. **The effect of using uniform vs. log-uniform bin distributions.** Best results in bold. While for the $[0.001, 80m]$ range of KITTI the log-uniform distribution is better, the  10m range of NYUv2 means that many log-uniformly distributed bins are clustered in a very small space. This may explain the drop in performance with NYUv2.

| Bins | Dset | Abs | RMS | RMSL | log10 | $\delta_1$ | $\delta_2$ | $\delta_3$ |
|---|---|---|---|---|---|---|---|---|
| 7 | nyu | 0.312 | 0.991 | 0.142 | 0.130 | 0.449 | 0.767 | 0.922 |
| 20 | nyu | $\underline{0.308}$ | $\underline{0.968}$ | $\underline{0.136}$ | $\underline{0.127}$ | $\underline{0.466}$ | $\underline{0.780}$ | $\underline{0.926}$ |
| 256 | nyu | **0.298** | **0.933** | **0.127** | **0.122** | **0.485** | **0.796** | **0.934** |
| 7 | kitti | $\underline{0.265}$ | 6.933 | 0.112 | 0.108 | 0.580 | 0.838 | 0.939 |
| 20 | kitti | $\underline{0.265}$ | $\underline{6.153}$ | $\underline{0.103}$ | $\underline{0.102}$ | $\underline{0.606}$ | $\underline{0.853}$ | $\underline{0.948}$ |
| 256 | kitti | **0.238** | **5.756** | **0.088** | **0.092** | **0.652** | **0.877** | **0.957** |

Table 7. **Effect of varying the number of bins used**, with learnable depth tokens. Bins are log-uniformly distributed. Best results in bold, second-best underlined. A clear correlation between bin number and performance may be seen across all metrics.

## 4.4. Varying Number And Distribution of Bins

The use of learned depth tokens removes the restriction on bin number: human-language ordinal scales require increasingly tortuous and verbose phrasing as bin granularity increases, but learnable depth tokens do not have this restriction. We also experiment with different bin distributions. As discussed in section 2.1, the distribution of depth values in an image is rarely linear, and the occlusion of the background by objects in the foreground naturally lends to a long-tailed distribution of depth values.

Table 6 compares the use of linearly and log-linearly distributed bins for different numbers of bins. We continue to use fixed rather than adaptive bin distributions to reduce trainable parameters and to better allow comparison. Table 6 shows that the log-uniform bin distribution is clearly better in the larger depth range of KITTI ($[0.001, 80m]$), but for NYUv2 it is not as clear. This is likely due to NYUv2's smaller range: log-uniformly-distributed bins are compressed into a very small space at the start of the range, and may not be sufficiently distinct to be useful.

Table 7 shows the effects of increasing the number of bins (and thus the number of depth-bin prompts), with log-uniform bin distribution. This adds 512 learnable parameters per bin, but a clear improvement in performance can be seen as the number of bins is increased.

## 4.5. Interpretation of Learned Tokens

Our results show that, for conveying the concept of specific depths to CLIP, the learned tokens are superior to the human-language depth and prompt context tokens. In this

| **{depth_0}** | | **{depth_1}** | | **{depth_2}** | | **{depth_3}** | | **{depth_4}** | | **{depth_5}** | | **{depth_6}** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Token | Dist. | Token | Dist. | Token | Dist. | Token | Dist. | Token | Dist. | Token | Dist. | Token | Dist. |
| *{depth_2}* | 0.177 | *close</w>* | 0.000 | *{depth_0}* | 0.177 | *{depth_2}* | 0.313 | *{depth_6}* | 0.185 | *distant</w>* | 0.000 | *{depth_4}* | 0.185 |
| *{depth_3}* | 0.320 | *closest</w>* | 0.725 | *{depth_3}* | 0.313 | *{depth_0}* | 0.320 | *{depth_2}* | 0.780 | *distance</w>* | 0.656 | *{depth_2}* | 0.776 |
| *{depth_6}* | 0.781 | *close* | 0.746 | *{depth_6}* | 0.776 | *{depth_6}* | 0.813 | *{depth_0}* | 0.790 | *dissi* | 0.907 | *{depth_0}* | 0.781 |
| *{depth_4}* | 0.790 | *closes</w>* | 0.835 | *{depth_4}* | 0.780 | *{depth_4}* | 0.820 | *{depth_3}* | 0.820 | *dist* | 0.924 | *{depth_3}* | 0.813 |
| *·</w>* | 0.895 | *clo* | 0.872 | *·</w>* | 0.907 | *coscino</w>* | 0.915 | *coscino</w>* | 0.913 | *thest</w>* | 0.977 | *coscino</w>* | 0.894 |
| *coscino</w>* | 0.911 | *glou* | 0.883 | *coscino</w>* | 0.918 | *·</w>* | 0.918 | *·</w>* | 0.976 | *ssian</w>* | 1.006 | *·</w>* | 0.955 |
| *atility</w>* | 0.923 | *closer</w>* | 0.887 | *atility</w>* | 0.928 | *atility</w>* | 0.918 | *atility</w>* | 0.981 | *pist* | 1.010 | *atility</w>* | 0.971 |
| *mikequind* | 0.949 | *closing</w>* | 0.887 | *mikequind* | 0.956 | *mikequind* | 0.969 | *mikequind* | 0.992 | *dian* | 1.013 | *mikequind* | 0.980 |
| *arty* | 0.979 | *chose</w>* | 0.914 | *arty* | 0.980 | *arty* | 0.982 | *laghate* | 1.010 | *drifting</w>* | 1.015 | *laghate* | 0.992 |
| *kirstel</w>* | 0.987 | *lose</w>* | 0.926 | *kirstel</w>* | 0.993 | *âł* | 0.994 | *ison</w>* | 1.018 | *distribu* | 1.022 | *ison</w>* | 1.000 |
| *rhea</w>* | 0.993 | *closed</w>* | 0.942 | *rhea</w>* | 1.002 | *ison</w>* | 0.995 | *kirstel</w>* | 1.023 | *distri* | 1.022 | *kirstel</w>* | 1.015 |
| *laghate* | 1.001 | *chosen</w>* | 0.947 | *ison</w>* | 1.005 | *kirstel</w>* | 1.002 | *rectan* | 1.042 | *titan* | 1.028 | *arty* | 1.015 |
| *ison</w>* | 1.002 | *choose* | 0.953 | *laghate* | 1.009 | *rhea</w>* | 1.005 | *arty* | 1.042 | *dis* | 1.033 | *rectan* | 1.026 |
| *âł* | 1.005 | *most</w>* | 0.958 | *pknot</w>* | 1.014 | *pknot</w>* | 1.013 | *soyuz</w>* | 1.055 | *disappear</w>* | 1.033 | *rhea</w>* | 1.029 |
| *pknot</w>* | 1.005 | *chooses</w>* | 0.971 | *âł* | 1.022 | *laghate* | 1.017 | *rhea</w>* | 1.056 | *dito</w>* | 1.034 | *soyuz</w>* | 1.030 |

Table 8. **Nearest-neighbours to learned depth tokens in CLIP embedding space.** Learned tokens from 7 evenly-distributed bins on NYUv2 using 'baseline' template from table 1. 'Distance' is Euclidean distance after normalisation of embeddings. Token 0 corresponds to a bin centre of approx. 0.714m, and token 6 to approx. 9.29m. We see that tokens 1 and 6 correspond with the tokens 'close</w>' and 'distant</w>' respectively, but that the remaining tokens are closest to other learned tokens.

section, we attempt to interpret these learned tokens, in particular their correspondence to human language.

The nearest-neighbours in CLIP-embedding space for each of the learned tokens are measured. Each of the 40k tokens in the CLIP vocabulary are inserted into the sentence "This object appears to be {token}." and embedded into CLIP space and the resulting set of embeddings is normalised. The nearest neighbours by Euclidean distance for each of the learned depth tokens is found and is shown in table 8. The same procedure was repeated in token space instead of embedding space, but no apparent relationship between learned tokens and relevant English words was observed. This is expected: token features are constrained to be semantically related, but tokens themselves are not.

Several interesting observations may be made from table 8. Firstly, the nearest neighbour for every learned token bar two is another learned token, indicating that they all occupy a similar region of latent space. This likely indicates shared meaning, though it may also be due to their shared initialisation distribution. Secondly, the learned depth tokens appear to exist in a continuum of a sort: the larger-valued tokens are nearest to other large-valued tokens, and similarly with smaller-valued tokens. This may mean that the learned tokens exist on a path through latent space, the distance along which may map to a variation in the concept of depth, e.g. nearby to distant. Thirdly, only two tokens correlate with depth-related words, with the rest relating to one another followed by unrelated tokens. This is the most relevant finding, as it indicates that while there is some relationship between the CLIP conception of depth and the human concepts of "close" and "distant", it also indicates that there is another, inexpressible, concept contained within the learnable depth tokens that is nonetheless important. This is further supported by the increase in performance from adding additional bins: a degenerate case where only two tokens contain meaning and the rest are irrelevant would yield similar performance regardless of the number of bins.

## 5. Conclusion and Future Work

In this work, we have illustrated the efficacy of learnable tokens for monocular depth estimation using CLIP. We see that with only a few thousand learnable parameters, we are able to extract significant performance improvements over the use of geometrically-related human-language words and random tokens. Analysis of the learned tokens shows that their CLIP embeddings are dissimilar to those of the depth-related human language prompts tried, suggesting that the concept of depth is not succinctly expressible in language.

All but two of the learned depth tokens are nearest in latent space to one another and not near any relevant English words. The remaining two are nearby words relating to the concepts of "close" and "distant", but a degenerate case where all but two tokens are ignored is unlikely given that performance continues to improve when more learnable tokens are introduced. This finding suggests that succinct human-language prompts that seem relevant to depth to a human writer do not, in fact, map to the region of CLIP's latent space that relates to CLIP's concept of depth.

The implications of this for any method that uses CLIP are clear: CLIP contains surprising general knowledge, but accessing it using human-chosen prompts may be sub-optimal, implying that its understanding of the world extends beyond the limits of what language can succinctly represent. While we focus on the concept of specific depths, it may be the case that other similarly abstract concepts could also require learnable, non-linguistic tokens to effectively describe them.

Future work will further investigate the CLIP embedding space, both in translating complex geometric concepts into tokens that CLIP can comprehend, and in learning to interpret CLIP features in a human-readable way, in order to enhance the utility of CLIP for all open-ended tasks.

## References

[1] Ashutosh Agarwal and Chetan Arora. Attention Attention Everywhere: Monocular Depth Prediction with Skip Attention, Oct. 2022. arXiv:2210.09071 [cs].

[2] Yucai Bai, Lei Fan, Ziyu Pan, and Long Chen. Monocular Outdoor Semantic Mapping with a Multi-task Network. *arXiv pre-print*, Jan. 2019. arXiv: 1901.05807.

[3] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. AdaBins: Depth Estimation using Adaptive Bins. *arXiv:2011.14141 [cs]*, Nov. 2020. arXiv: 2011.14141.

[4] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. LocalBins: Improving Depth Estimation by Learning Local Distributions, Mar. 2022. arXiv:2203.15132 [cs].

[5] Fabian Deuser, Konrad Habel, Philipp J. Rösch, and Norbert Oswald. Less Is More: Linear Layers on CLIP Features as Powerful VizWiz Model, June 2022. arXiv:2206.05281 [cs].

[6] David Eigen and Rob Fergus. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture. In *ICCV*, 2015.

[7] David Eigen, Christian Puhrsch, and Rob Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. In *NIPS*, 2014.

[8] Ravi Garg, Vijay Kumar B G, Gustavo Carneiro, and Ian Reid. Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue. In *ECCV*, 2016. arXiv: 1603.04992v2.

[9] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[10] Jiaxian Guo, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Boyang Li, Dacheng Tao, and Steven C. H. Hoi. From Images to Textual Prompts: Zero-shot VQA with Frozen Large Language Models, Dec. 2022. arXiv:2212.10846 [cs].

[11] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual Prompt Tuning, July 2022. arXiv:2203.12119 [cs].

[12] Jianbo Jiao, Ying Cao, Yibing Song, and Rynson Lau. Look Deeper into Depth: Monocular Depth Estimation with Semantic Booster and Attention-Driven Loss. In *ECCV*, 2018.

[13] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From Big to Small: Multi-Scale Local Planar Guidance for Monocular Depth Estimation. *arXiv:1907.10326 [cs]*, Aug. 2019. arXiv: 1907.10326.

[14] Yi Li, Hualiang Wang, Yiqun Duan, Hang Xu, and Xiaomeng Li. Exploring Visual Interpretability for Contrastive Language-Image Pre-training, Nov. 2022. arXiv:2209.07046 [cs].

[15] Zhenyu Li, Xuyang Wang, Xianming Liu, and Junjun Jiang. BinsFormer: Revisiting Adaptive Bins for Monocular Depth Estimation. page 21, 2022.

[16] Ron Mokady, Amir Hertz, and Amit H. Bermano. ClipCap: CLIP Prefix for Image Captioning, Nov. 2021. arXiv:2111.09734 [cs].

[17] Jia Ning, Chen Li, Zheng Zhang, Zigang Geng, Qi Dai, Kun He, and Han Hu. All in Tokens: Unifying Output Space of Visual Tasks via Soft Token, Jan. 2023. arXiv:2301.02229 [cs].

[18] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural Discrete Representation Learning, May 2018. arXiv:1711.00937 [cs].

[19] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *arXiv:2103.00020 [cs]*, Feb. 2021. arXiv: 2103.00020.

[20] Michaël Ramamonjisoa and Vincent Lepetit. SharpNet: Fast and Accurate Recovery of Occluding Contours in Monocular Depth Estimation. *arXiv preprint*, 2019. arXiv: 1905.08598v1.

[21] Sheng Shen, Liunian Harold Li, Hao Tan, Mohit Bansal, Anna Rohrbach, Kai-Wei Chang, Zhewei Yao, and Kurt Keutzer. How Much Can CLIP Benefit Vision-and-Language Tasks?, July 2021. arXiv:2107.06383 [cs].

[22] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Empirical Methods in Natural Language Processing (EMNLP)*. arXiv, Nov. 2020. arXiv:2010.15980 [cs].

[23] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*, 2012.

[24] Haoyu Song, Li Dong, Wei-Nan Zhang, Ting Liu, and Furu Wei. CLIP Models are Few-shot Learners: Empirical Studies on VQA and Visual Entailment, Mar. 2022. arXiv:2203.07190 [cs].

[25] Lijun Wang, Jianming Zhang, Oliver Wang, Zhe Lin, and Huchuan Lu. SDC-Depth: Semantic Divide-and-Conquer Network for Monocular Depth Estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 538–547, Seattle, WA, USA, June 2020. IEEE.

[26] Daitao Xing, Jinglin Shen, Chiuman Ho, and Anthony Tzes. ROIFormer: Semantic-Aware Region of Interest Transformer for Efficient Self-Supervised Monocular Depth Estimation, Dec. 2022. arXiv:2212.05729 [cs].

[27] Renrui Zhang, Ziyao Zeng, and Ziyu Guo. Can Language Understand Depth? In *ACM Multimedia 2022*. arXiv, July 2022. arXiv:2207.01077 [cs].

[28] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional Prompt Learning for Vision-Language Models, Oct. 2022. arXiv:2203.05557 [cs].

[29] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to Prompt for Vision-Language Models. *International Journal of Computer Vision*, 130(9):2337–2348, Sept. 2022. arXiv:2109.01134 [cs].