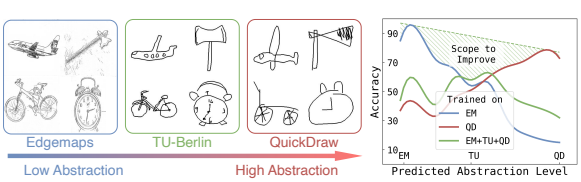
Do Generalised Classifiers *really work* on Human Drawn Sketches?

Hmrishav Bandyopadhyay Pinaki Nath Chowdhury Aneeshan Sain
Subhadeep Koley Tao Xiang Ayan Kumar Bhunia Yi-Zhe Song

SketchX, CVSSP, University of Surrey
{h.bandyopadhyay, p.chowdhury, a.sain,
s.koley, t.xiang, a.bhunia, y.song}@surrey.ac.uk



Naively training CLIP + prompt learning on QuickDraw (QD), or TU-Berlin (TU) sketches, or Edgemaps (EM) fails to generalise to multiple abstraction levels.

Training	Evaluation	CLIP	Ours
QD	QD+TU+EM	43.24	47.35 (↑ 4.1)
TU	QD+TU+EM	43.71	51.03 (↑ 7.3)
EM	QD+TU+EM	42.91	43.76 (↑ 0.8)
QD+TU+EM	QD+TU+EM	45.59	62.96 (↑ **17.4**)

Fig. 1: Unlike photos, sketch classification poses additional challenges such as *abstraction* – humans draw differently based on their subjective interpretations, sketching ability, and drawing time. Existing datasets such as TU-Berlin [23] and QuickDraw [32] only capture the time axis and collect human sketches drawn under 280 and 20 seconds, respectively. Following [23,32,34,71] we consider Edgemaps (**EM**) as low abstract, TU-Berlin (**TU**) sketches as medium abstract, and QuickDraw (**QD**) ones as highly abstract (*left*). Naively training CLIP via prompt learning, [92] on sketches (*right*) from one abstraction level (**QD**, **TU**, or **EM**) individually do not generalise across varying abstractions (**QD** + **TU** + **EM**). Jointly training on multiple abstractions (**QD** + **TU** + **EM**) is also sub-optimal (45.6 on CoOp [92] vs 62.9 on Ours). (*middle*) Importantly, our proposed method predicts a classification score and an abstraction level for input sketches. Plotting classification accuracy vs predicted abstraction level, reveals a *scope for improvement* (shaded region) showing – despite our significant improvement in classification (↑ 17.4%) over naive CLIP + prompt learning, generalisation across varying sketch abstractions is still an open problem. We hope this will motivate future works to democratise existing methods [52, 91] for human drawn sketches.

Abstract. This paper, for the first time, marries large foundation models with human sketch understanding. We demonstrate what this brings – a paradigm shift in terms of generalised sketch representation learning (e.g., classification). This generalisation happens on two fronts: (i) generalisation across unknown categories (i.e., open-set), and (ii) generalisation traversing abstraction levels (i.e., good and bad sketches), both being timely challenges that remain unsolved in the sketch literature. Our design is intuitive and centred around transferring the already stellar generalisation ability of CLIP to benefit generalised learning for sketches. We first "condition" the vanilla CLIP model by learning sketch-specific prompts using a novel auxiliary head of raster to vector sketch conversion. This importantly makes CLIP "sketch-aware". We then make CLIP acute to the inherently different sketch abstraction levels. This

is achieved by learning a codebook of abstraction-specific prompt biases, a weighted combination of which facilitates the representation of sketches across abstraction levels – low abstract edge-maps, medium abstract sketches in TU-Berlin, and highly abstract doodles in QuickDraw. Our framework surpasses popular sketch representation learning algorithms in both zero-shot and few-shot setups and in novel settings across different abstraction boundaries.

Keywords: Sketch Classification · Sketch Abstraction · Zero-Shot

1 Introduction

The vision community is witnessing a paradigm shift in the face of large foundation models [52, 55]. Instead of learning visual semantics from scratch, the rich semantics inherent to foundation models are explored to enrich visual learning, as in [3, 24, 40] for retrieval, and [14, 44, 74] for generation. The most salient advantage that foundation models bring is their generalisation ability [27,39,91,92], which made a significant impact on zero-shot and few-shot learning.

In this paper, we marry human sketches with foundation models to naturally tackle two of most significant bottlenecks in the sketch community, with little effort, piggybacking on generalisability of foundation models. First is the data scarcity problem of human drawn sketches – the largest sketch dataset (QuickDraw [32]) contains 350 categories compared with the easily > 1000 categories for photos [18], which makes generalised learning for sketch even more pronounced a problem. The second is, although everyone *can* sketch, most people sketch *differently* as a result of varying drawing skills and diverse subjective interpretations – see Fig. 1 for distinctly different sketches of a "bike". These sketch-specific challenges call for a single model generalising along two axes – (i) across *unseen categories* for the first data scarcity challenge, and (ii) across *abstraction levels* for the second challenge of sketches exhibiting varying abstraction levels.

Solving these challenges, we show that it all comes down to making CLIP [52] sketch-specific. For the former (data scarcity problem), we learn a set of continuous vectors (visual prompts) injected into CLIP's vision transformer encoder. This enables CLIP (pre-trained on $\sim$400M image-text pairs) to adapt to sketches while preserving its generalisation ability – resulting in a generalised sketch recogniser that works across unknown categories. More specifically, we first design two sets of *visual prompts* – shallow prompts injected into the initial layer of the CLIP transformer and deep prompts injected up to a depth of 9 layers. Keeping rest of CLIP frozen, we train these prompts on the extracted sketch and `[category]` embeddings (as class labels) from CLIP text encoder using cross-entropy loss. Although shallow+deep prompts encourage CLIP to become "sketch-aware", they do not model any sketch-specific traits during training. Hence, we additionally use an auxiliary task of raster to vector sketch conversion that exploits the dual representation of sketch [7] to reinforce that awareness.

The latter challenge of dealing with sketch abstraction[1] is less obvious and extends the status quo of what is possible with foundation models. While there is no consensus on *what constitutes an abstract sketch* [17], prior works follow: (i) number of strokes (more stroke → less abstract) [71], and (ii) sketch drawing time (more time → less abstract) [23, 32]. Since precisely annotating the abstraction score for each sketch is ill-posed, we learn sketch abstraction semi-supervised. Particularly, we assign semi-accurate, coarse abstraction levels as: (i) doodles from QuickDraw dataset [32] drawn under 20 seconds as *high abstraction*, (ii) freehand sketches from TU-Berlin [23] drawn under 280 seconds as *medium abstraction*, and (iii) Edgemaps from [87] as *low abstraction*. To model the continuous abstraction spectrum (low → high), we employ codebook-learning [47]. Our abstraction codebook comprises of three "codes" (a learned feature embedding), where the continuous abstraction is modelled by mixing (a weighted average) the discrete codes. To make CLIP "abstraction-aware", the abstraction vector (from mixing codes) is injected as an additional prompt to the generalised sketch classifier. To our best knowledge, ours is the first work showing the potential of combining codebook learning [47] and prompt learning [93] for generalised sketch representation learning.

In summary, our contributions are: (i) we, for the first time marry human sketches with foundation models to tackle two of the most significant bottlenecks facing the sketch community – data scarcity and abstraction levels. (ii) For data scarcity, we achieve generalisation across unseen categories by adapting CLIP for sketch classification via prompt learning. (iii) To further make CLIP "sketch-aware", we exploit sketch-specific traits like raster-to-vector sketch conversion as an auxiliary loss. (iv) For abstraction, we achieve generalisation across varying abstraction levels using a codebook-driven approach, where a mixup of learned codebook vectors acts as prompts that interface with CLIP, with CLIP, to make our model robust to recognising sketches from multiple abstraction levels, including those not seen during training.

2 Related Works

Sketch for Visual Understanding: Sketches can depict visual concepts [34] using intuitive free-hand drawings, overcoming linguistic barriers often faced in text representations. Fine-grained in nature, a sketch is an attractive query medium for tasks like sketch-based image [15, 21, 58, 65] and 3D shape retrieval [78]. Creative sketches [28] encouraged sketch-based synthesis and editing of images [43, 83, 85], natural objects or scenes [13, 26, 73], faces [12], and animation frames [82]. Despite being expressed as monochrome lines on a 2D plane, sketches convey complex 3D structures and find use in 3D shape modelling [5, 31, 79, 90]. As an interactive medium , a sketch is an important modality of input in vision tasks like sketch-based object detection [68], image-inpainting [83], representation learning [72], incremental learning [9], image-segmentation [35], etc. Beyond its discriminative [58] or representative [72] potential, sketch has also

[1] For rest of the paper, we denote 'abstraction' as $\mathbb{A}$.

been employed for pictionary style gaming [8]. The success of sketch-based visual understanding leads us to propose a framework for recognising any (open-set [4]) free-hand drawing on unseen categories.

Sketch Classification: Early approaches to sketch understanding [64] extract hand-crafted features like shape primitives [49], bag-of-words [23], or Fischer Vectors [61] from raster (static pixel-map) sketches. Better representations were formulated with deep learning algorithms, dramatically improving sketch classification on complete [60,62] and partial sketches [63], surpassing even humans [84] in recognition. Alternate architectural designs encode the temporal order of vector strokes (pen coordinates & motions) [32] using RNNs [32] or Transformers [54]. Fusion of both raster representations for encoding structural information and vector representations for temporal cues has been employed with CNN-RNN feature fusion [41] for million-scale recognition with hashing [76], or with Graph Neural Networks [51, 67, 77, 81]. Various sketch-specific self-supervised methods have recently emerged that employ BERT-like [19] pre-training [42], jigsaw solving [48], or cross-modal translation between raster-vector dual representation of sketches [7]. While recognition in existing works is limited to seen classes, we, for the first time, introduce a zero-shot recognition pipeline that can recognise unseen classes under varying sketch abstraction levels.

CLIP in Vision Tasks: Contrastive Language-Image Pre-training (CLIP) [52] pairs rich semantics from text-descriptions with large-scale (∼400M) image datasets, exploiting underlying relationships in cross-modal data (image+text) by representing them in the same embedding space. As such, CLIP features are highly generalisable in downstream tasks using very-few (few-shot) or no labels (zero-shot) [92] as opposed to traditional features (trained on discrete labels). This adaptation to downstream tasks like few-shot recognition [27], retrieval [3,25,56], object detection [30], semantic segmentation [53], image generation [14], etc., is done primarily through prompting, first introduced for NLP [10]. Prompting, in general, involves construction of a task-specific template (e.g., `'[MASK] is the capital of the France'`), which is then filled with word labels (e.g. `'London/Paris'`). Prompts give context to the word labels, forming an appropriate text description. Engineering prompts by hand requires domain expertise, hence recent works (prompt tuning) model them as learnable continuous vectors [91, 92] to be optimised directly during fine-tuning. Beyond language prompts, visual prompts in the form of continuous vectors have also been explored in visual feature extractors like ViT [36]. In contrast to learning general [2, 36, 92] or instance-specific prompts [91], we learn continuous abstraction prompts modelled on abstraction level of input sketch. This enables the recognition of a wide range of sketches, from professional edge-map like drawings to free-hand abstract doodles [32].

Abstraction in Sketches: Sketch abstraction ($\mathbb{A}$) was first defined as a factor of time [6] with an observation that users tend to draw only salient regions in a constrained time setting. This idea was later modelled as a trade-off between compactness and recognisability [45, 46] for the 'generation' of abstract strokes by removing strokes based on their salience. Parametric representations

model abstraction through Bézier curves, [16] and differential equations [17], controlling stroke "complexity" with Bézier control points and sinusoidal frequencies respectively. Alternate representations of abstraction include modelling sketch as a combination of appearance and structure [80] in a coarse-to-fine manner through hierarchical feature encoder learning [57], or via a composition of predefined drawing primitives [1] to identify the most distinctive parts of the sketch, and ground them into interpretable features. In contrast, we represent abstraction on a continuous spectrum, varying from Edgemaps (equivalent to professional sketches) to human-drawn doodles (highly abstract) by learning abstraction-specific codebook vectors, which we interpolate in a zero/few-shot sketch recognition setup.

3 Background

CLIP: The generalisability of CLIP makes it a popular choice [56] for open-set vision-language tasks. Specifically, CLIP uses 2 independent encoders: *(i)* a ResNet [33] or Vision Transformer [20] image encoder, and (ii) a transformer-based [69] text encoder. The Vision Transformer image encoder F processes input images as r fixed-sized patches $\mathrm{I} = \{p_1, \ldots, p_r\};\ p_j \in \mathbb{R}^{3\times h\times w}$ that are embedded along with an extra learnable class token [19] c^p. These are then passed through transformer layers [69] with multi-head attention to obtain the visual features $f_p = F(I, c^p) \in \mathbb{R}^d$. The text input (say, n words) is pre-processed to word-embeddings $W_0 = \{\mathbf{w}_0^j\}_{j=1}^n$, from which the text transformer T extracts textual features as $t_0 = \mathrm{T}(W_0) \in \mathbb{R}^d$. Since CLIP maps cross-modal (image+text) features on the same embedding space, features of text-photo pairs have maximal similarity $\texttt{sim}(\cdot)$ compared to features from unpaired (mismatched) samples after contrastive training. For zero-shot classification, textual prompts like '`a photo of a [category]`' (from a list of K categories) are used to obtain category-specific textual features $\{t_j\}_{j=1}^K$. The probability of input photo I (with photo-feature f_p) belonging to y^{th} category can be calculated as

$$\mathbb{P}(y|\mathrm{I}) = \frac{\exp\left(\texttt{sim}\left(f_p, t_y\right)/\tau\right)}{\sum_{j=1}^{K}\exp\left(\texttt{sim}\left(f_p, t_j\right)/\tau\right)} \tag{1}$$

Prompt Learning: Prompt learning uses foundation models, like BERT [19] and CLIP [52], as knowledge bases from which useful information can be extracted for downstream tasks [50]. Apart from using handcrafted prompts like '`a photo of a`', recent methods like CoOp [92] and CoCoOp [91] learns n continuous context vectors, $\{v_1, \ldots, v_n\}$, each having $v_j \in \mathbb{R}^{d_t}$ dimension word embeddings. With base CLIP frozen, the continuous context vectors v_j are learned by backpropagating gradients through the text $\mathrm{T}(\cdot)$ encoder. Using word embedding for the k-th '`[category]`', denoted as $w_0^k \in \mathbb{R}^{d_t}$, the prompt is constructed as $[v_1, \ldots, v_n, w_0^k]$, and passed to the transformer to obtain text feature $f_t = \mathrm{T}([v_1, \ldots, v_n, w_0^k])$. Recent works learn continuous context vectors (i.e., prompts) for text [92] and image [2] multimodal prompts [39]. Variations of

prompt learning also include shallow vs. deep [75] prompts, depending on the layers where the context vectors are injected. We use deep prompt learning to adapt CLIP to design a generalised sketch classifier.

4 Proposed Methodology

In this paper, we build a generalised sketch classifier that works in an unseen setup. This "unseen" problem in sketch representation learning has two axes: (a) generalisation across unseen categories – train on 'cats' or 'dogs' (not on 'zebras') but evaluate on 'zebras'; and (b) generalisation across abstractions – a sketch can be drawn in 20 seconds (i.e., highly abstract doodle) or 280 seconds (TU-Berlin sketches [23]). For generalisation across categories, we use the open-vocabulary potential of CLIP [52], which has excellent generalisation across several downstream tasks. Particularly, we show how off-the-shelf CLIP is sub-optimal, and a simple yet significant sketch-specific adaptation with prompt learning [93] and raster-to-vector self-reconstruction objective [7] can help generalisation to unseen categories. Generalising across abstraction ($\mathbb{A}$) levels is challenging as $\mathbb{A}$ is hard-to-label and a *subjective metric* (e.g., it is hard quantifying "how abstract is that sketch" on a scale 0 to 1.0). Hence, we propose a weakly-supervised codebook-learning paradigm [47] to learn generalisation across sketch abstractions.

4.1 Generalisation Across Unseen Categories

Baseline Sketch Classifier: Sketch classification aims at predicting the category a given query-sketch belongs to. An input raster sketch $\mathrm{I}_S \in \mathbb{R}^{3\times H\times W}$ is encoded using a backbone feature extractor $f_s = \mathrm{F}(\mathrm{I}) \in \mathbb{R}^d$ like ResNet-101 [33] followed by mapping it to a K-dimensional vector $\mathrm{F}_c : \mathbb{R}^d \rightarrow \mathbb{R}^K$ that classifies I_S into predefined K categories $f_c = \mathrm{F}_c(f_s) \in \mathbb{R}^K$. Both backbone $\mathrm{F}(\cdot)$ and classifier $\mathrm{F}_c(\cdot)$ are learned given ground-truth class $\hat{f}_{c,k}$ as,

$$\mathcal{L}_{\mathrm{CE}} = -\hat{f}_{c,k} \log \frac{\exp(f_{c,k})}{\sum_{j=1}^{K} \exp(f_{c,j})} \tag{2}$$

Prompt Learning to Adapt CLIP for Sketches: We use CLIP with ViT visual encoder [52] which extends the fixed set classifier F_c in Eq. (2) into an open-set setup. We now learn J sketch prompts $\mathbf{v^s} = \{v_0^s, \ldots, v_{J-1}^s\}$, where $v_j^s \in \mathbb{R}^{5\times d_p}$. First, we divide the raster sketch into r fixed-sized patches $\mathrm{I}_S = \{s_1, \ldots, s_r\}$ where each patch $s_i \in \mathbb{R}^{3\times h\times w}$ is embedded as $E_0 = \{e_0^j\}_{j=1}^r$. Next, the learnable prompts are injected into each transformer block of CLIP vision transformer $\mathrm{F}(\cdot)$ up to a specific depth J, as

$$\begin{aligned} [c_j^p, E_j, _\,] &= \mathrm{F}_j([c_{j-1}^p, E_{j-1}, v_{j-1}^s])\ |_{j=1}^{J} \\ [c_i^p, E_i, v_i^s] &= \mathrm{F}_i([c_{i-1}^p, E_{i-1}, v_{i-1}^s])\ |_{i=J+1}^{N} \\ f_s &= \texttt{ImageProj}(c_N^p) \end{aligned} \tag{3}$$

where, c_0^p is a pre-trained `[CLS]` token (see Sec. 3). To classify the visual feature $f_s \in \mathbb{R}^d$, we use handcrafted prompts like '`a photo of a [category]`' that is encoded using CLIP text encoder $\mathrm{T}(\cdot)$ into f_t as in Eq. (1). However, (i) our input is '`sketch`' not '`photo`', and (ii) handcrafted prompts are sub-optimal compared to learnable prompts [92], $\mathbf{v^t} = \{v_0^t, \ldots, v_{J-1}^t\}$; $v_j^t \in \mathbb{R}^{5 \times d_t}$. Hence, we inject prompts $\mathbf{v^t}$ in $\mathrm{T}(\cdot)$ up to depth J as,

$$\begin{aligned} [_, w_j^k] &= \mathrm{T}_j([v_{j-1}^t, w_{j-1}^k]) \; |_{j=1}^{J} \\ [v_i^t, w_i^k] &= \mathrm{T}_i([v_{i-1}^t, w_{i-1}^k]) \; |_{i=J+1}^{N} \\ f_t &= \texttt{TextProj}(w_N^k) \end{aligned} \tag{4}$$

where, w_0^k is the word embedding of '`[category]`'. Naively using learnable prompts $\mathbf{v^t}$ overfits to training/seen categories, lacking generalisation to unseen categories [91]. Hence, we use a lightweight Meta-Net $\pi = \mathrm{H}(f_s)$ to predict an instance-specific context, $\pi \in \mathbb{R}^{5 \times d_t}$ that shifts $\mathbf{v^t}$ as $\mathbf{v^t}(f_s) = \{v_0^t + \pi, \ldots, v_{J-1}^t + \pi\}$. Intuitively, Meta-Net (a two-layer Linear-ReLU-Linear) reduces overfitting of $\mathbf{v^t}$ to training/seen categories, generalising better for unseen classes using sketch-conditional prompts $\mathbf{v^t}(f_s)$. Finally,

$$\mathcal{L}_{\mathrm{CE}} = -\log \frac{\exp\left(\texttt{sim}\left(f_s, \mathrm{T}([\mathbf{v^t}(f_s), w_0^i])\right)/\tau\right)}{\sum_{j=1}^{K} \exp\left(\texttt{sim}\left(f_s, \mathrm{T}([\mathbf{v^t}(f_s), w_0^j])\right)/\tau\right)} \tag{5}$$

Auxiliary Loss using Sketch Specific Traits: Sketches are uniquely characterised by its existence in dual modalities – rasterised images $\mathrm{I}_S \in \mathbb{R}^{3 \times H \times W}$ and vector coordinate sequences $\mathrm{I}_V \in \mathbb{R}^{N \times 5}$. Translating $\mathrm{I}_S \rightarrow \mathrm{I}_V$ enforces image encoder $\mathrm{F}(\cdot)$, particularly its learnable prompts $\mathbf{v^s}$, to learn sparse stroke information. Accordingly, we use a linear embedding layer [7] to obtain the initial hidden state of a Gated Recurrent Unit (GRU) decoder as $h_0 = W_h f_s + b_h$. Its hidden state h_t is updated as: $h_t = \texttt{GRU}(h_{t-1}; [f_s, P_{t-1}])$, where P_{t-1} is the last predicted point. Next, an embedding layer is used to predict a five-element vector at each time step as $P_t = W_p h_t + b_p$ where $P_t = (x_t, y_t, q_t^1, q_t^2, q_t^3) \in \mathbb{R}^{2+3}$ whose first two logits represent absolute coordinate (x, y) and the later three for pen's state position (q^1, q^2, q^3). We use mean-square error and categorical cross-entropy loss to train raster $\rightarrow$ vector on ground-truth absolute coordinates $(\hat{x}_t, \hat{y}_t)$ and pen state $(\hat{q}_1, \hat{q}_2, \hat{q}_3)$ as,

$$\begin{aligned} \mathcal{L}_{\mathrm{S} \rightarrow \mathrm{V}} = \frac{1}{T} \sum_{t=1}^{N} ||\hat{x}_t - x_t||_2 + ||\hat{y}_t - y_t||_2 \\ - \frac{1}{N} \sum_{t=1}^{N} \sum_{i=3}^{3} \hat{q}_t^i \log\left(\frac{\exp(q_t^i)}{\sum_{j=1}^{3} \exp(q_t^j)}\right) \end{aligned} \tag{6}$$

4.2 Generalisation Across Sketch Abstractions

Pilot Study: Here we elaborate Fig. 1 (*right*) that examines generalisation of CLIP when abstractions vary from **EM** $\rightarrow$ **TU** $\rightarrow$ **QD**. We randomly select 40 classes common across **QD**, **TU** and **EM** – 20 seen classes to adapt CLIP via

prompt learning (CoOp [92]), and 20 unseen classes for zero-shot evaluation. We observe: (i) training and evaluating on the same abstraction (**QD**, or **TU**, or **EM**) performs ∼5.09% better, than training on one and jointly evaluating on **QD** + **TU** + **EM**. This drop signifies that a naive CLIP + prompt learning fails to generalise across abstractions. (ii) Jointly training on sketches from **QD** + **TU** + **EM**, only marginally improves accuracy by 2.30%. This highlights an *even deeper* problem – simply scaling the training data will *likely*[2] not solve the *varying* sketch abstraction problem.

Overview: Although we can easily sketch at varying abstraction levels, collecting precise abstraction annotation is difficult. Fig. 1 (*left*), shows that *in general*, **QD** doodles are more abstract than **TU** sketches, however precisely annotating abstraction score from 0 → 1 for every sample (e.g., "bike" in **QD**) is an ill-posed problem. Prior works developed *proxy metrics* for sketch abstraction, like the number of strokes (fewer strokes ⇒ higher abstraction) [71], drawing skills (amateur vs. professional) [29], and time to sketch (lesser time ⇒ higher abstraction) [59]. Instead, we design our method based on the general consensus [23, 32, 34, 59] that – (i) **EM**s are (usually) less abstract than **TU**, and (ii) **QD** doodles are (usually) more abstract than **TU**. Particularly, while **QD** + **TU** + **EM** do not cover all possible sketch abstractions, **EM** and **QD** are a good approximation of the *lower* and *upper* bounds of sketch abstractions.

Abstraction ($\mathbb{A}$) Learning without Annotations: Although drawing a sketch at varying abstractions is easy, annotating its precise abstraction level is hard, inferring the need of a weakly-supervised approach for abstraction modelling. Given that **EM** → **QD** roughly provides a low→high abstraction [23, 32, 34, 59], we define a classification problem: represent **QD** (high $\mathbb{A}$) by ground-truth class label $\hat{\mathbb{A}}_h$, **TU** (medium $\mathbb{A}$) by $\hat{\mathbb{A}}_m$, and **EM** (low $\mathbb{A}$) by $\hat{\mathbb{A}}_l$. For each abstraction level, we learn a codebook vector $\{\theta_l, \theta_m, \theta_h\}$ where $\theta_{l,m,h} \in \mathbb{R}^{5\times d_t}$. Given an input sketch I_S, our backbone sketch encoder extracts $f_s = \mathrm{F}(\mathrm{I}_S, \mathbf{v^s})$, which is then fed into a codebook classifier $\mathcal{C}_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^3$ to get a `softmax` normalised probability distribution over the three abstraction class labels as, $[\mathbb{A}_l, \mathbb{A}_m, \mathbb{A}_h] = \mathcal{C}_\theta(f_s)$. We train $\mathcal{C}_\theta(\cdot)$ via a categorical cross-entropy loss as,

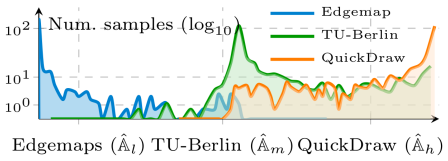


Fig. 2: Plotting number of sketches vs class membership of 600 sketch instances, defined by class-labels $(\hat{\mathbb{A}}_l, \hat{\mathbb{A}}_m, \hat{\mathbb{A}}_h)$ and softmax normalised distributions $(\mathbb{A}_l, \mathbb{A}_m, \mathbb{A}_h)$. Sketches are taken from 20 unseen categories common across **QD**, **TU**, and **EM** with 10 sketches per category. Despite expected peaks, a significant number of sketches lie in the continuous spectrum between $\hat{\mathbb{A}}_l \rightarrow \hat{\mathbb{A}}_m$ and $\hat{\mathbb{A}}_m \rightarrow \hat{\mathbb{A}}_h$ (overlaps).

$$\mathcal{L}_{\text{CB}} = -(\hat{\mathbb{A}}_l \log \mathbb{A}_l + \hat{\mathbb{A}}_m \log \mathbb{A}_m + \hat{\mathbb{A}}_h \log \mathbb{A}_h) \tag{7}$$

[2] Exploring scaling laws [37] for human sketch abstraction is an interesting and non-trivial problem for future work.

The predicted scores are used to combine (weighted summation) learned codebooks as $\eta = \mathbb{A}_l\theta_l + \mathbb{A}_m\theta_m + \mathbb{A}_h\theta_h$ which acts as an abstraction prompt $\eta \in \mathbb{R}^{5\times d_t}$ and shifts the sketch-conditional prompt $\mathbf{v^t}(f_s)$ (Sec. 4.1) like a bias as, $\mathbf{v^t}(f_s^*) = \{(v_0^1 + \pi + \eta), \ldots, (v_{J-1}^t + \pi + \eta)\}$. Next, we use Eq. (5) to classify. **Augmentations using Abstraction-Mixup:** Eq. (7) helps us model abstractions in human drawn sketch by learning a codebook vector for 3-levels $[\theta_l, \theta_m, \theta_h]$ and predicting their `softmax` normalised probabilities $[\mathbb{A}_l, \mathbb{A}_m, \mathbb{A}_h]$. Unseen sketches however do not adhere to these predefined levels and are often on a continuous *spectrum* of abstraction ($\mathbb{A}_{l\leftrightarrow h}$). To verify this, we compute the class membership among **EM**, **TU** and **QD** for 600 sketches using the softmax normalised distribution as: $\mathbb{A}_l$ of class ($\hat{\mathbb{A}}_l$), $\mathbb{A}_m$ of class ($\hat{\mathbb{A}}_m$), and $\mathbb{A}_h$ of class ($\hat{\mathbb{A}}_h$). These 600 sketches are taken from the unseen split of 20 categories common across **EM**, **TU** and **QD**, where each category has 10 sketches. From Fig. 2, while there is an expected peak near class labels ($\hat{\mathbb{A}}_l, \hat{\mathbb{A}}_m, \hat{\mathbb{A}}_h$), we observe: (i) a significant number of sketches lie in the continuous spectrum between $\hat{\mathbb{A}}_l \leftrightarrow \hat{\mathbb{A}}_m$ and $\hat{\mathbb{A}}_m \leftrightarrow \hat{\mathbb{A}}_h$. This indicates that sketch abstraction is *not discrete* at $\hat{\mathbb{A}}_l, \hat{\mathbb{A}}_m, \hat{\mathbb{A}}_h$ but *continuous* from **EM** $\leftrightarrow$ **QD**. (ii) The abstraction of sketches in **TU** vary widely, overlapping with those in **QD** and **EM**. Assigning all sketches in **TU** (class $\hat{\mathbb{A}}_m$) to a fixed discrete level modelled by θ_m can corrupt generalisation [86]. To alleviate this hard assumption, we propose *abstraction-mixup* – a simple extension of an augmentation routine, *mixup* [70, 86]. Now, we randomly sample sketches $\{\mathrm{I}_S^l, \mathrm{I}_S^m, \mathrm{I}_S^h\}$ from {**QD**, **TU**, **EM**} and obtain $\{f_s^l, f_s^m, f_s^h\}$ (using $\mathrm{F}(\cdot)$) respectively. Next, we randomly sample the mixing coefficients from a 3-dimensional Dirichlet distribution as $\{\lambda_1, \lambda_2, \lambda_3\} \sim \mathtt{Dir}(\alpha)$ where,

$$\mathtt{Dir}(\lambda_1,\lambda_2,\lambda_3;\alpha) = \frac{\Gamma(3\alpha)}{\Gamma(\alpha)^3}\prod_{i=1}^{3}\lambda_i^{\alpha-1} \tag{8}$$

and $\Gamma(\cdot)$ is the gamma function with $\alpha > 0$. Next, we compute a *mixup sketch feature* $f_s^\alpha = \lambda_1^* f_s^l + \lambda_2^* f_s^m + \lambda_3^* f_s^h$, where $\lambda_i^* = \lambda_i / (\sum_{j=1}^3 \lambda_j)$ is the normalised coefficients. Using f_s^α, we train the codebook classifier $[\mathbb{A}_l^\alpha, \mathbb{A}_m^\alpha, \mathbb{A}_h^\alpha] = \mathcal{C}_\theta(f_s^\alpha)$ by modifying Eq. (7) as,

$$\mathcal{L}_{\text{mix}} = -(\lambda_1^* \log \mathbb{A}_l^\alpha + \lambda_2^* \log \mathbb{A}_m^\alpha + \lambda_3^* \log \mathbb{A}_h^\alpha) \tag{9}$$

Essentially, $\mathcal{L}_{\text{mix}}$ helps to augment synthetic latent representations of sketches across a continuous spectrum of sketch abstraction. Our final training loss combines Eqs. (5) to (7) and (9) with coefficients (hyper-parameters) $\beta_{1,2,3}$ as,

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{CE}} + \beta_1\, \mathcal{L}_{\text{S}\rightarrow\text{V}} + \beta_2\, \mathcal{L}_{\text{CB}} + \beta_3\, \mathcal{L}_{\text{mix}} \tag{10}$$

4.3 Inference Pipleline

First, we use CLIP vision transformer $\mathrm{F}(\cdot)$ and our learned sketch prompts $\mathbf{v^s} = \{v_0^s, \ldots, v_{J-1}^s\}$, where $v_j^s \in \mathbb{R}^{5\times d_p}$, to encode an input sketch I_S into

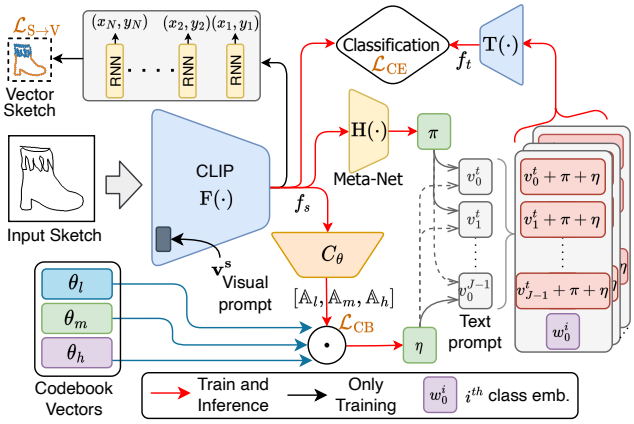


Fig. 3: Given an input sketch, we compute visual feature f_s with sketch prompts using a CLIP image encoder. Next, f_s is fed into 4 pipelines: (i) An auxiliary raster-to-vector (sketch2vec) translation module that distils sketch-specific traits. (ii) A Meta-Net to predict an instance-specific context π to generalise on unseen sketches. (iii) A codebook classifier $\mathcal{C}_\theta$ to compute an abstraction prompt η, given codebook vectors $\{\theta_l, \theta_m, \theta_h\}$. (iv) Combine text prompts $\mathbf{v^t}$ with π, η, and category embedding w_0^k to generate text feature f_t using CLIP text encoder T(·). Finally, class-specific text features f_t are matched with f_s to compute class probabilities.

a visual feature $f_s = \mathrm{F}(\mathrm{I}_S; \mathbf{v^s}) \in \mathbb{R}^d$. **Second**, f_s is simultaneously given to two modules: (i) a lightweight Meta-Net to predict instance-specific context, $\pi = \mathrm{H}(f_s)$, where $\pi \in \mathbb{R}^{5\times d_t}$, and (ii) a codebook classifier $\mathcal{C}_\theta : \mathbb{R}^d \to \mathbb{R}^3$ to get `softmax` normalised probability distribution over the three abstraction class labels, $[\mathbb{A}_l, \mathbb{A}_m, \mathbb{A}_h] = \mathcal{C}_\theta(f_s)$. The predicted scores are used to get the abstraction prompt $\eta = \mathbb{A}_l\theta_l + \mathbb{A}_m\theta_m + \mathbb{A}_h\theta_h$, where $\eta \in \mathbb{R}^{5\times d_t}$, and $\{\theta_l, \theta_m, \theta_h\}$ are the codebook vectors. **Third**, for classification, we compute the word embedding for K categories as $\{w_0^k\}_{k=1}^K$. **Finally**, we concatenate word embedding w_0^k to the sum of our learned text prompt $\mathbf{v^t}$, instance-specific context π, and abstraction prompt η to obtain the final text feature f_t using CLIP text encoder as $f_t = \mathrm{T}(\ [(\mathbf{v^t} + \pi + \eta), w_0^k]\ ) \in \mathbb{R}^d$. Classification probability is calculated from Eq. (1).

5 Experiments

Implementation Details: We use pre-trained CLIP with ViT-B/16 (vision transformer) as visual encoder F(·) and transformer-based text encoder T(·). For text encoder, we learn five 512-dimensional context vectors as prompt $v_j^t \in \mathbb{R}^{5\times 512}$. The class token for the k-th '`[category]`' is given by $w_0^k \in \mathbb{R}^{512}$. We learn five 768-dimensional sketch prompts $v_j^s \in \mathbb{R}^{5\times 768}$. The learnable prompts are injected upto a depth $J = 9$, where $\{v_0^s, v_0^t\}$ are *shallow* prompts, while the *deep* prompts are $\{v_1^s, \ldots, v_{J-1}^s\} \in \mathbb{R}^{8\times 5\times 512}$ and $\{v_1^t, \ldots, v_{J-1}^t\} \in \mathbb{R}^{8\times 5\times 512}$ for vision and text encoders, respectively. Although CLIP's weights are frozen during training, we fine-tune the layer-norm parameters for improved performance [56]. Our method consisting of Meta-Net + Codebooks + layer-norm + vision prompts ($\mathbf{v^s}$) + text prompts ($\mathbf{v^t}$) is trained with Adam optimizer for 7 epochs with $1e{-}4$ learning rate and 64 batch-size.

Datasets: We use sketches from QuickDraw [32] and TU-Berlin [23] along with Edgemaps [11] of TU-Berlin Extended [87]. Ranked from highest to lowest abstraction, QuickDraw has 50M sketches across 345 categories, TU-Berlin has 20K sketches from 250 categories, and the Edgemaps are generated using [11] from 204K images across 250 categories in TU-Berlin extended. For few-shot training, we randomly pick 10 sketches per class from a list of 125 classes common to all three datasets and reserve the remaining ones (220 for QuickDraw, 125 for TU-Berlin, and 125 for Edgemaps) for zero-shot inference. Generating Edgemaps from complex scene images (with noisy backgrounds) leads to noisy sketches. We filter images with high classification scores (higher score ⇒ less noisier background) using pre-trained zero-shot CLIP (details in supplementary).

Evaluation Setup: We evaluate our algorithm on two fronts: *(i) few-shot accuracy*: where we train our model on 10 randomly sampled sketches from each of 125 common classes in all three datasets and evaluate them on previously unseen samples from the same class list. *(ii) zero-shot accuracy*: where we use our previously trained few-shot model and evaluate on unseen samples from new classes in these datasets. This difficult evaluation setup helps us understand *(a)* how well the model generalises to unseen classes i.e., how much did we *adapt* the generalisation potential of CLIP for sketch recognition, and *(b)* how well the model trained on seen categories, generalises across varying abstractions using codebook vectors and their mix-up. We also evaluate the adaptability of our network, by replacing CLIP-backbone with FLAVA [66].

Methods	EM		TU [23]		QD [32]	
	Seen	Unseen	Seen	Unseen	Seen	Unseen
CLIP-Z [52]	52.09	50.10	56.57	47.71	20.00	13.27
CoOp [92]	55.06	50.66	58.92	47.92	22.80	12.64
CoCoOp [91]	56.03	51.57	59.74	50.25	24.48	12.68
VPT-A [2]	52.82	41.22	66.08	51.01	37.02	15.36
MaPLe [39]	61.01	52.90	71.66	53.91	36.24	17.74
Linear Probe [52]	34.84	–	57.24	–	36.94	–
Tip-Adapter [88]	60.58	–	65.74	–	42.24	–
Sketch-A-Net [84]	–	–	27.01	3.14	18.08	0.68
ResNet [33]	9.00	2.16	18.34	1.82	7.20	0.63
ResNet-Adapt	8.68	1.33	25.18	2.33	9.84	0.36
Edge-Augment	14.95	0.70	35.17	0.81	9.52	0.46
VPT-P (Shallow) [36]	27.92	–	46.56	- -	24.33	–
VPT-P (Deep) [36]	42.08	–	55.83	–	34.00	–
Ours	**66.72**	**59.05**	**76.96**	**60.51**	**45.20**	**22.41**

Table 1: Recognition accuracy for sketches in **EM**, **TU**, and **QD**. All competitors are jointly trained on **QD** + **TU** + **EM**.

Prompt Depth	Context Token	Meta Net	Layer Norm	Codebook Vectors	Mixup	Sketch2 Vec	Accuracy Seen	Accuracy Unseen
1	5	✓	✓	✓	✓	✓	66.87	59.39
3	5	✓	✓	✓	✓	✓	69.42	58.99
7	5	✓	✓	✓	✓	✓	74.52	57.77
9	2	✓	✓	✓	✓	✓	74.52	60.61
9	10	✓	✓	✓	✓	✓	74.21	57.56
9	20	✓	✓	✓	✓	✓	75.54	55.94
9	5	✗	✓	✓	✓	✓	73.41	54.19
9	5	✓	✗	✓	✓	✓	73.50	59.90
9	5	✓	✓	✗	✗	✓	73.09	60.20
9	5	✓	✓	✓	✗	✓	71.36	60.00
9	5	✓	✓	✓	✓	✗	73.70	59.49
9	5	✓	✓	✓	✓	✓	**76.96**	**60.51**

Table 2: Ablation studies on TU-Berlin [23]: With varying *Prompt Depth* and *number of Context Tokens* trained with/without *LayerNorm fine-tuning*,*Meta-Net*, *Codebook Vectors*, *auxiliary Sketch2Vec* and *Mixup*.

Competitors: We compare against *(i)* existing state-of-the-art (SOTA) zero-shot and few-shot recognition methods. **CLIP** [52] classifies sketches by comparing a sketch encoding from the visual encoder with class encodings from the text encoder using hand-crafted text prompts like '`a photo of a [category]`'. **CoOp** [92] extends CLIP by replacing hand-crafted prompts with learnable text prompts. **VPT-A** [2] learns a visual prompt instead, that is added to the image to adapt the vision encoder for sketch classification via hand-crafted text prompts (similar to *CLIP*). **MaPLe** [39] learns a joint vision-text "deep prompt" inserted in multiple layers of the vision and text encoders for better sketch and class encodings respectively. We use the independent vision-language prompting mode of *MaPLe* for a fairer comparison. Contrary to prior works learning static prompts, **CoCoOp** [91] learns an adaptive text prompt – a bias vector π

conditioned on the input sketch and added to the learned text prompt in *CoOp*. **Linear Probe** [52] classifies sketch by adding a linear layer at the end of CLIP's visual encoder. **Tip-Adapter** [89] uses a CLIP-based non-parametric query-key cache model [38] as an adapter with similarity-based retrieval to determine the class of test sample from its feature encoding. *(ii)* Apart from adapting CLIP-based SOTAs, we provide a comprehensive comparison with widely used sketch classifiers like **ResNet** [33] and **Sketch-A-Net** [84]. **ResNet-Adapt** bridges the domain gap between abstract sketches in **TU** [23], **QD** [32], and **EM** using a domain discriminator to align ResNet visual features from all three domains. **Edge-Augment** [22] fine-tunes a ResNet, pre-trained on geometrically augmented Edgemaps, on real sketches. **VPT-P** [36] learns visual prompts for the vision transformer [20], injected as image patches in **VPT-P (Shallow)**, or multiple deeper layers [39] in **VPT-P (Deep)**.

5.1 Sketch Recognition

We report few-shot and zero-shot recognition results of our algorithm on **QD**, **TU**, and **EM** in Tab. 1 using average accuracy across all datasets for reference. **Few-Shot Recognition:** We obtain a Top-1 accuracy of 66.72%, 76.96%, and 45.20% on **EM**, **TU**, and **QD** respectively with our algorithm, beating SOTA *MaPLe* by an average margin of 6.66%. Works on shallow language prompts like *CoOp* (45.59%) and *CoCoOp* (46.75%) and shallow visual prompts like *VPT-A* (51.97%) beat *zero-shot CLIP* by an average of 2.71%, 3.87%, and 9.09%, reinforcing the idea of prompt learning for better recognition. We note that visual prompting on CLIP (*VPT-A*) yields better results than language prompting (*CoCoOp*) in a shallow prompt setting, with an accuracy difference of 5.22%. As analysed in [39] and evident from our experiments, we note that deeper prompts like those in *MaPLe* (56.33%) and *VPT-P (Deep)* (43.97%) outperform their shallow prompt counterparts like *VPT-P (shallow)* (32.93%). Our work also outperforms few-shot methods that use hand-crafted prompts like *Tip-Adapter* [88] and *Linear Probe* [52] by 6.91% and 20.09% respectively. The superiority of our abstraction handling algorithm is demonstrated by its performance-*gain* of 48.53% over naive adaptation-based solutions like *ResNet-Adapt* (14.56%) for multi-dataset training. *Sketch-A-Net*, having a sketch-specific network architecture, beats *ResNet/ResNet-Adapt* pre-trained on images by 8.67/1.83% and 10.88/8.24% on **TU** and **QD**, respectively. As *Sketch-A-Net* requires stroke order information, we do not report accuracy against **EM** that lack this data.
Zero-Shot Recognition: We find zero-shot performance in CLIP-based methods to be significantly higher than baseline networks, as CLIP is pre-trained on 400M image-text pairs, making it easy for CLIP to recognise sketch categories unseen during fine-tuning. Zero-shot CLIP beats non-CLIP baselines like *Sketch-A-Net* and *ResNet-Adapt* by as much as 29.27% and 35.69%, respectively. Amongst CLIP-based methods, we find our method tto outperform *MaPLe* [39] marginally (5.80%) and *zero-shot CLIP* considerably (10.30%) in recognition accuracy. CLIP-based methods that are adapted specifically for few-shot training (*Tip-Adapter* [88] and *Linear Probe* [52]) are not reported under Zero-Shot

Recognition. Furthermore, replacing the CLIP-backbone with FLAVA [66], we find our model improves upon zero-shot recognition with FLAVA (38.37/35.21 %) by +17.36/+9.49 % for categories seen/unseen by our model.

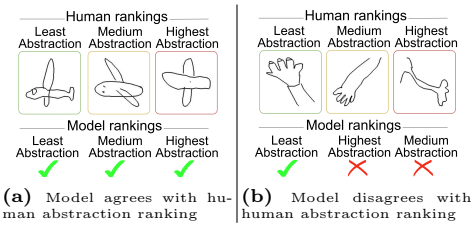
5.2 Ablation

We ablate various components and hyperparameters in Tab. 2. *(i)* Varying prompt depth J affects recognition accuracy, where $J = 1$ (shallow prompts) drops it to 66.87% and 59.39% in seen and unseen classes, respectively. *(ii)* Varying length of language prompt ($v_j^t \in \mathbb{R}^{5\times512}$) from 5 to 2, 10 and 20 drops accuracy to 74.52%, 74.21% and 75.54% respectively. *(ii)* Removing the MetaNet drops accuracy by 4.94%, particularly zero-shot accuracy by 6.32%. *(iii)* Removing $\mathcal{L}_{\text{S}\to\text{V}}$ drops accuracy by 3.26%, due to lack of sketch-specific traits. *(iv)* Removing codebook learning that models abstraction drops accuracy by 3.87%. *(v)* Removing codebook mixup that models continuous abstraction drops accuracy by 3.06%. *(vi)* Removing layer-norm drops accuracy by 2.03%.

5.3 Recognising the Degree of Abstraction

The codebook classifier predicts abstraction score as a softmax normalised probability distribution on 3 coarse abstraction levels: $\hat{\mathbb{A}}_l$ (**EM**), $\hat{\mathbb{A}}_m$ (**TU**), $\hat{\mathbb{A}}_h$ (**QD**)[3].

Quantitative: Given unseen sketches from **QD**, **TU**, or **EM**, we predict its abstraction score as a softmax normalised distribution over codebooks $\{\mathbb{A}_l, \mathbb{A}_m, \mathbb{A}_h\}$. Our model effectively classifies unseen sketches into their ground-truth abstraction level $\{\hat{\mathbb{A}}_l, \hat{\mathbb{A}}_m, \hat{\mathbb{A}}_h\}$ with an average accuracy of 95%. To verify that our abstraction classification is not merely a sketch dataset classification, we perform a human study, where we perform a detailed analysis of human rankings for abstraction levels $(0, 0.5, 1.0)$ against our predicted abstraction ranks.



(a) Model agrees with human abstraction ranking **(b)** Model disagrees with human abstraction ranking

Fig. 4: Human study to rank 3 sketches from same category and same dataset into low, medium, or high abstraction levels.

Human Study: Sketch abstraction is a subjective metric and hard to quantify. As prior works have no consensus on "*what is sketch abstraction*" [17, 23, 32, 71], we conduct a human study, where we select 20 common categories across 3 datasets. Next from each dataset, and each category we select 3 sketches (*e.g.*, 3 bicycle sketches all from TU-Berlin), giving us a total of 20 (categories) $\times$ 3 (datasets) $= 60$ triplets.

We then engage 5 users across a diverse demographic in the age range of $20-30$. We provide a set of 12 triplets (a triplet has 3 sketches per category per dataset) and ask them to rank each sketch (Fig. 4) by its abstraction in the triplet.

[3] Since there is no precise abstraction annotation for each sketch, we assign coarse abstraction levels to **EM**, **TU**, and **QD**.

Accordingly, the ranked-list of 60 sketch triplets is compared with our model's prediction using the abstraction classifier. Our predicted ranking aligns with humans at an avg. correlation of 71.67%, 78.33%, and 66.67% for low, medium, and high abstraction levels respectively (Sec. 5.3).

Next, we shuffle the 5 sets of 12 triplets among the 5 users, such that no one receives their old set. Now, for each triplet, we show participants our predicted abstraction ranking and ask them to rate their overall agreement $(1-5)$ to the provided ranking (Fig. 5) with 1 being "Strongly Disagree" and 5 being "Strongly Agree". For a total of 60 triplets, users report an average agreement score (Mean Opinion Score) of 4.3.

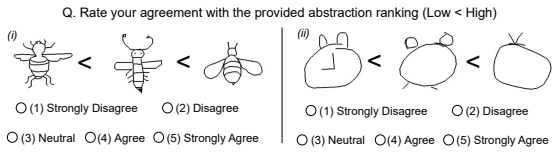


Fig. 5: User opinion on abstraction rankings

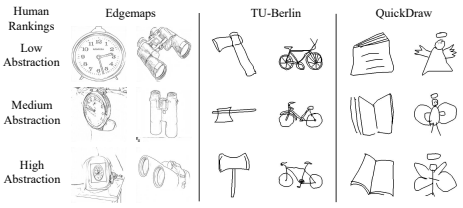


Fig. 6: Human rankings of sketch triplets.

Human Study	EM	TU [23]	QD [32]
Rank-1	13 (65 %)	15 (75 %)	12 (60 %)
Rank-2	16 (80 %)	17 (85 %)	15 (75 %)
Rank-3	14 (70 %)	15 (75 %)	13 (65 %)

Table 3: No. of sketches on which our network agrees with the humans on most abstract (Rank-1), average (Rank-2), and least abstract (Rank-3).

High correlation and Mean Opinion Scores, even when sketch triplets come from the same dataset, proves our codebook classifier is not simply a dataset classifier.

6 Conclusion

We extend the notion of a generalised classifier from photos to sketches. Towards this goal, we adapt CLIP (with open-set generalisation) for sketches by learning prompts for both the vision and language branches. In addition, to learn sketch-specific traits, we employ an auxiliary raster $\rightarrow$ vector sketch reconstruction loss. Finally, we generalise CLIP across varying sketch abstractions. As sketches lack precise abstraction annotation, we assign coarse-level scores to Edgemaps as low abstraction, TU-Berlin sketches as medium, and QuickDraw doodles as high abstraction. We employ codebook learning and *mixup* to learn sketch abstraction in a semi-supervised setup. The resulting SketchCLIP serves as a foundation model for robust sketch recognition algorithms based on large-scale vision-language models to classify *"any"* abstract sketch.

References

1. Alaniz, S., Mancini, M., Dutta, A., Marcos, D., Akata, Z.: Abstracting Sketches through Simple Primitives. In: ECCV (2022)
2. Bahng, H., Jahanian, A., Sankaranarayanan, S., Isola, P.: Exploring Visual Prompts for Adapting Large-Scale Models. arXiv preprint arXiv:2203.17274 (2022)

3. Baldrati, A., Bertini, M., Uricchio, T., Del Bimbo, A.: Effective Conditioned and Composed Image Retrieval Combining CLIP-Based Features. In: CVPR (2022)
4. Bendale, A., Boult, T.: TowardsOpenWorldRecognition. In: CVPR (2015)
5. Berardi, G., Gryaditskaya, Y.: Fine-Tuned but Zero-Shot 3D Shape Sketch View Similarity and Retrieval. In: ICCV SHARP Workshop (ICCV) (2023)
6. Berger, I., Shamir, A., Mahler, M., Carter, E., Hodgins, J.: Style and abstraction in portrait sketching. ACM TOG (2013)
7. Bhunia, A.K., Chowdhury, P.N., Yang, Y., Hospedales, T.M., Xiang, T., Song, Y.Z.: Vectorization and Rasterization: Self-Supervised Learning for Sketch and Handwriting. In: CVPR (2021)
8. Bhunia, A.K., Das, A., Muhammad, U.R., Yang, Y., Hospedales, T.M., Xiang, T., Gryaditskaya, Y., Song, Y.Z.: Pixelor: A Competitive Sketching AI Agent. So you think you can sketch? ACM TOG (2020)
9. Bhunia, A.K., Gajjala, V.R., Koley, S., Kundu, R., Sain, A., Xiang, T., Song, Y.Z.: Doodle it yourself: Class incremental learning by drawing a few sketches. In: CVPR (2022)
10. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language Models are Few-Shot Learners. In: NeurIPS (2020)
11. Chan, C., Durand, F., Isola, P.: Learning to generate line drawings that convey geometry and semantics. In: CVPR (2022)
12. Chen, S.Y., Su, W., Gao, L., Xia, S., Fu, H.: DeepFaceDrawing: Deep generation of face images from sketches. ACM TOG (2020)
13. Chen, W., Hays, J.: Sketchygan: Towards diverse and realistic sketch to image synthesis. In: ICCV (2018)
14. Chen, Z., Wang, G., Liu, Z.: Text2Light: Zero-Shot Text-Driven HDR Panorama Generation. ACM TOG (2022)
15. Collomosse, J., Bui, T., Jin, H.: Livesketch: Query perturbations for guided sketch-based visual search. In: CVPR (2019)
16. Das, A., Yang, Y., Hospedales, T., Xiang, T., Song, Y.Z.: BézierSketch: A generative model for scalable vector sketches. In: ECCV (2020)
17. Das, A., Yang, Y., Hospedales, T., Xiang, T., Song, Y.Z.: SketchODE: Learning neural sketch representation in continuous time. In: ICLR (2021)
18. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
19. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: NAACL (2019)
20. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: ICLR (2021)
21. Dutta, A., Akata, Z.: Semantically tied paired cycle consistency for zero-shot sketch-based image retrieval. In: CVPR (2019)
22. Efthymiadis, N., Tolias, G., Chum, O.: Edge Augmentation for Large-Scale Sketch Recognition without Sketches. In: ICPR (2022)
23. Eitz, M., Hays, J., Alexa, M.: How Do Humans Sketch Objects? ACM TOG (2012)
24. Fang, H., Xiong, P., Xu, L., Chen, Y.: Clip2video: Mastering video-text retrieval via image clip. arXiv preprint arXiv:2106.11097 (2021)

25. Fang, K., Song, J., Gao, L., Zeng, P., Cheng, Z.Q., Li, X., Shen, H.T.: Pros: Prompting-to-simulate generalized knowledge for universal cross-domain retrieval. In: CVPR (2024)
26. Gao, C., Liu, Q., Xu, Q., Wang, L., Liu, J., Zou, C.: Sketchycoco: Image generation from freehand scene sketches. In: CVPR (2020)
27. Gao, P., Geng, S., Zhang, R., Ma, T., Fang, R., Zhang, Y., Li, H., Qiao, Y.: CLIP-Adapter: Better Vision-Language Models with Feature Adapters. IJCV (2023)
28. Ge, S., Goswami, V., Zitnick, L., Parikh, D.: Creative Sketch Generation. In: ICLR (2021)
29. Gryaditskaya, Y., Sypesteyn, M., Hoftijzer, J.W., Pont, S., Durand, F., Bousseau, A.: OpenSketch: A Richly-Annotated Dataset of Product Design Sketches. ACM SIGGRAPH (2019)
30. Gu, X., Lin, T.Y., Kuo, W., Cui, Y.: Open-vocabulary Object Detection via Vision and Language Knowledge Distillation. In: ICLR (2022)
31. Guillard, B., Remelli, E., Yvernay, P., Fua, P.: Sketch2Mesh: Reconstructing and Editing 3D Shapes from Sketches. In: CVPR (2021)
32. Ha, D., Eck, D.: A Neural Representation of Sketch Drawings. In: ICLR (2018)
33. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: ICCV (2016)
34. Hertzmann, A.: Why do line drawings work? A realism hypothesis. Perception (2020)
35. Hu, C., Li, D., Yang, Y., Hospedales, T.M., Song, Y.Z.: Sketch-a-segmenter: Sketch-based photo segmenter generation. IEEE TIP (2020)
36. Jia, M., Tang, L., Chen, B.C., Cardie, C., Belongie, S., Hariharan, B., Lim, S.N.: Visual prompt tuning. In: ECCV (2022)
37. Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling Laws for Neural Language Models. arXiv preprint arXiv:2001.08361 (2020)
38. Khandelwal, U., Levy, O., Jurafsky, Zettlemoyer, L., Lewis, M.: Generalization through Memorization: Nearest Neighbor Language Models. In: ICLR (2020)
39. Khattak, M.U., Rasheed, H., Maaz, M., Khan, S., Khan, F.S.: MaPLe: Multi-modal Prompt Learning. In: CVPR (2023)
40. Lei, J., Li, L., Zhou, L., Gan, Z., Berg, T.L., Bansal, M., Liu, J.: Less is more: Clipbert for video-and-language learning via sparse sampling. In: CVPR (2021)
41. Li, H., Jiang, X., Guan, B., Wang, R., Thalmann, N.M.: Multistage Spatio-Temporal Networks for Robust Sketch Recognition. IEEE TIP (2022)
42. Lin, H., Fu, Y., Xue, X., Jiang, Y.G.: Sketch-bert: Learning sketch bidirectional encoder representation from transformers by self-supervised learning of sketch gestalt. In: CVPR (2020)
43. Liu, H., Wan, Z., Huang, W., Song, Y., Han, X., Liao, J., Jiang, B., Liu, W.: Deflocnet: Deep image editing via flexible low-level controls. In: CVPR (2021)
44. Mirowski, P., Banarse, D., Malinowski, M., Osindero, S., Fernando, C.: Clip-clop: Clip-guided collage and photomontage. arXiv preprint arXiv:2205.03146 (2022)
45. Muhammad, U.R., Yang, Y., Hospedales, T.M., Xiang, T., Song, Y.Z.: Goal-driven sequential data abstraction. In: CVPR (2019)
46. Muhammad, U.R., Yang, Y., Song, Y.Z., Xiang, T., Hospedales, T.M.: Learning deep sketch abstraction. In: ICCV (2018)
47. Oord, A.v.d., Vinyals, O., Kavukcuoglu, K.: NeuralDiscreteRepresentationLearning. In: NeurIPS (2017)
48. Pang, K., Yang, Y., Hospedales, T.M., Xiang, T., Song, Y.Z.: Solving Mixed-modal Jigsaw Puzzle for Fine-Grained Sketch-Based Image Retrieval. In: CVPR (2020)

49. Paulson, B., Hammond, T.: PaleoSketch: Accurate Primitive Sketch Recognition and Beautification. In: IUI (2008)
50. Petroni, F., Rockt aschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A.H., Riedel, S.: Language models as knowledge bases? In: EMNLP (2019)
51. Qi, Y., Su, G., Chowdhury, P.N., Li, M., Song, Y.Z.: SketchLattice: Latticed Representation for Sketch Manipulation. In: ICCV (2021)
52. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML (2021)
53. Rao, Y., Zhao, W., Chen, G., Tang, Y., Zhu, Z., Huang, G., Zhou, J., Lu, J.: DenseCLIP: Language-Guided Dense Prediction with Context-Aware Prompting. In: CVPR (2022)
54. Ribeiro, L.S.F., Bui, T., Collomosse, J., Ponti, M.: Sketchformer: Transformer-based representation for sketched structure. In: CVPR (2020)
55. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-Resolution Image Synthesis with Latent Diffusion Models. In: CVPR (2022)
56. Sain, A., Bhunia, A.K., Chowdhury, P.N., Koley, S., Xiang, T., Song, Y.Z.: Clip for all things zero-shot sketch-based image retrieval, fine-grained or not. In: CVPR (2023)
57. Sain, A., Bhunia, A.K., Yang, Y., Xiang, T., Song, Y.Z.: Cross-Modal Hierarchical Modelling forFine-Grained Sketch Based Image Retrieval. In: BMVC (2020)
58. Sain, A., Bhunia, A.K., Yang, Y., Xiang, T., Song, Y.Z.: Stylemeup: Towards style-agnostic sketch-based image retrieval. In: CVPR (2021)
59. Sangkloy, P., Burnell, N., Ham, C., Hays, J.: The sketchy database: learning to retrieve badly drawn bunnies. ACM TOG (2016)
60. Sarvadevabhatla, R.K., Babu, R.V.: Freehand Sketch Recognition Using Deep Features. In: ICIP (2015)
61. Schneider, R.G., Tuytelaars, T.: Sketch classification and classification-driven analysis using fisher vectors. ACM TOG (2014)
62. Seddati, O., Dupont, S., Mahmoudi, S.: DeepSketch: Deep Convolutional Neural Networks for Sketch Recognition and Similarity Search. In: CBMI (2015)
63. Seddati, O., Dupont, S., Mahmoudi, S.: DeepSketch 2: Deep Convolutional Neural Networks for Partial Sketch Recognition. In: CBMI (2016)
64. Sezgin, T.M., Stahovich, T., Davis, R.: Sketch Based Interfaces: Early Processing for Sketch Understanding. In: PUI (2001)
65. Shen, Y., Liu, L., Shen, F., Shao, L.: Zero-shot sketch-image hashing. In: ICCV (2018)
66. Singh, A., Hu, R., Goswami, V., Couairon, G., Galuba, W., Rohrbach, M., Kiela, D.: Flava: A foundational language and vision alignment model. In: CVPR (2022)
67. Su, G., Qi, Y., Pang, K., Yang, J., Song, Y.Z.: Sketchhealer: A graph-to-sequence network for recreating partial human sketches. In: BMVC (2020)
68. Tripathi, A., Dani, R.R., Mishra, A., Chakraborty, A.: Sketch-guided object localization in natural images. In: ECCV (2020)
69. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
70. Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., Bengio, Y.: Manifold mixup: Better representations by interpolating hidden states. In: ICML (2019)
71. Vinker, Y., Pajouheshgar, E., Bo, J.Y., Bachmann, R.C., Bermano, A.H., Cohen-Or, D., Zamir, A., Shamir, A.: Clipasso: Semantically-aware object sketching. ACM TOG (2022)

72. Wang, A., Ren, M., Zemel, R.: Sketchembednet: Learning novel concepts by imitating drawings. In: ICML (2021)
73. Wang, S.Y., Bau, D., Zhu, J.Y.: Sketch your own gan. In: CVPR (2021)
74. Wang, Z., Liu, W., He, Q., Wu, X., Yi, Z.: Clip-gen: Language-free training of a text-to-image generator with clip. arXiv preprint arXiv:2203.00386 (2022)
75. Xing, Y., Wu, Q., Cheng, D., Zhang, S., Liang, G., Zhang, Y.: Class-aware visual prompt tuning for vision-language pre-trained model. arXiv preprint arXiv:2208.08340 (2022)
76. Xu, P., Huang, Y., Yuan, T., Pang, K., Song, Y.Z., Xiang, T., Hospedales, T.M., Ma, Z., Guo, J.: Sketchmate: Deep hashing for million-scale human sketch retrieval. In: ICCV (2018)
77. Xu, P., Joshi, C.K., Bresson, X.: Multi-Graph Transformer for Free-Hand Sketch Recognition. IEEE TNNLS (2022)
78. Xu, R., Han, Z., Hui, L., Qian, J., Xie, J.: Domain Disentangled Generative Adversarial Network for Zero-Shot Sketch-Based 3D Shape Retrieval. In: AAAI (2022)
79. Yan, G., Chen, Z., Yang, J., Wang, H.: Interactive liquid splash modeling by user sketches. ACM TOG (2020)
80. Yang, L., Pang, K., Zhang, H., Song, Y.Z.: Sketchaa: Abstract representation for abstract sketches. In: CVPR (2021)
81. Yang, L., Sain, A., Li, L., Qi, Y., Zhang, H., Song, Y.Z.: S3NET: Graph Representational Network For Sketch Recognition. In: ICME (2020)
82. Yi, R., Ye, Z., Fan, R., Shu, Y., Liu, Y.J., Lai, Y.K., Rosin, P.L.: Animating portrait line drawings from a single face photo and a speech signal. In: ACM SIGGRAPH (2022)
83. Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Free-form image inpainting with gated convolution. In: CVPR (2019)
84. Yu, Q., Yang, Y., Liu, F., Song, Y.Z., Xiang, T., Hospedales, T.M.: Sketch-a-net: A deep neural network that beats humans. IJCV (2017)
85. Zeng, Y., Lin, Z., Patel, V.M.: Sketchedit: Mask-free local image manipulation with partial sketches. In: CVPR (2022)
86. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond Empirical Risk Minimization. In: ICLR (2018)
87. Zhang, H., Liu, S., Zhang, C., Ren, W., Wang, R., Cao, X.: Sketchnet: Sketch classification with web images. In: ICCV (2016)
88. Zhang, R., Fang, R., Gao, P., Zhang, W., Li, K., Dai, J., Qiao, Y., Li, H.: Tip-Adapter: Training-free CLIP-Adapter for Better Vision-Language Modeling. In: ECCV (2022)
89. Zhang, R., Zhang, W., Fang, R., Gao, P., Li, K., Dai, J., Qiao, Y., Li, H.: Tip-Adapter: Training-free Adaption of CLIP for Few-shot Classification. In: ECCV (2022)
90. Zhang, S.H., Guo, Y.C., Gu, Q.W.: Sketch2Model: View-aware 3d modeling from single free-hand sketches. In: CVPR (2021)
91. Zhou, K., Yang, J., Loy, C.C., Liu, Z.: Conditional prompt learning for vision-language models. In: CVPR (2022)
92. Zhou, K., Yang, J., Loy, C.C., Liu, Z.: Learning to prompt for vision-language models. IJCV (2022)
93. Zhu, B., Niu, Y., Han, Y., Wu, Y., Zhang, H.: Prompt-aligned Gradient for Prompt Tuning. In: ICCV (2023)