Emergent Visual-Semantic Hierarchies in Image-Text Representations –Supplementary Material–

Morris Alper and Hadar Averbuch-Elor

Tel Aviv University

1 Interactive Visualization Tool

Please see our project page for an interactive visualization of model results on the *HierarCaps* test set. There we also provide visualizations of a random subset of the *HierarCaps* train set.

2 Additional Details

2.1 *HierarCaps* Construction

Below we provide a full technical description of the procedure used to construct *HierarCaps*. Examples of items from the train set of *HierarCaps*, along with the full test set, may be viewed in our interactive visualization tool.

Models and Inference Settings (*HierarCaps*) In this section, we describe the models used in producing *HierarCaps* and their inference settings.

To generate text for *HierarCaps*, we use the LLM Llama-2 [15] to produce hierarchies from existing captions in a zero-shot manner. We use the following model checkpoints (from the Hugging Face model hub):

- `meta-llama/Llama-2-13b-chat-hf` (henceforth LLM_{chat})
- `meta-llama/Llama-2-13b-hf` (henceforce LLM_{base})

We use the former instruction-tuned model for zero-shot tasks including instruction prompts (Section 2.1). We use the latter for few-shot augmentation of new synthetic hierarchies (Section 2.1). We generate text using stochastic sampling with temperature (1.2 for LLM_{chat} and 1.0 for LLM_{base}). In all cases, we load models using 4-bit quantization as implemented `bitsandbytes` integration with the Hugging Face `transformers` library, to enable inference on a single NVIDIA A5000 GPU.

In addition to generation, we also use a pretrained Natural Language Inference (NLI) model to filter for the desired logical properties (entailment or contradiction). In particular, we use BART-Large [7] fine-tuned on the MNLI

dataset [19], using the checkpoint `facebook/bart-large-mnli` from the Hugging Face model hub. When providing text to the NLI model, we use the prompt `a picture of: "{}"`, inserting the text into the slot indicated by `{}`.

For the smaller model $\mathcal{H}$ used in our knowledge distillation procedure, we use the encoder-decoder model Flan-T5 [3], using the `google/flan-t5-base` checkpoint from the Hugging Face model hub. Training details for this model are described in Section 2.1. When generating text with this model, we use stochastic sampling with temperature 1.0.

Seed Data Generation We begin by producing a small (∼900-item) set of caption hierarchies. This is motivated by the fact that we can produce these using real captions with a time-consuming and failure-prone process, but having these seeds will later allow us to produce hierarchies more reliably (see Sections 2.1–2.1).

To produce four-tiered caption hierarchies, we first select captions from Conceptual Captions [13] (CC) containing at least ten words, placing these in the fourth tier (T_4). We then apply prompts from Table 1 using LLM_{chat} (see Section 2.1) in order to produce preceding tiers; in other words, we apply the first prompt with the text of T_4 inserted in the blank slot (`{}`) to produce T_3, we then insert T_3 into the blank slot of the next prompt to produce T_2, and similarly we insert T_2 into the last prompt to produce T_1. Note that, in addition to the instruction delimiters (`[INST]` `[/INST]`), the prompts contain partial answers by design, as this enforces a uniform format for answers. LLM outputs are postprocessed by selecting only for text enclosed in quotation marks to avoid irrelevant text in outputs, and by removing texts that do not contain standard alphanumeric characters.

To ensure that these hierarchies have the desired logical structure, we filter each generated tier using the pretrained NLI model (described in Section 2.1). In particular, to each pair of tiers (T_i, T_j) with $i < j$ $(i, j \in \{1, 2, 3, 4\})$ we apply the NLI model to output probabilities $(p_e^{(ij)}, p_e^{(ji)})$, where $p_e^{(ij)}$ is the probability that T_i logically entails T_j, and $p_e^{(ji)}$ is the probability that T_j logically entails T_i. We filter so that for $i < j$, we have $p_e^{(ij)} < 0.5$ and $p_e^{(ji)} > 0.5$. In other words, we ensure that (up to the predictions of the NLI model) each tier is logically entailed by all following tiers, but does not entail them itself. Hence, each tier should be strictly more general than the following tiers.

Due to time and compute constraints, we run this process on a 5K-item subset of CC, yielding ∼900 seed hierarchies. Examples of such data include the following (displayed in the format $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4$):

- *cutting tools → scissors → middle-aged woman cutting white fabric with scissors in super slow motion → super slow motion of middle-age woman hands with scissors cutting white fabric on a tabletop*
- *dessert → pie → decorated pie on a white background → pie with candles and gifts in boxes on a white background*
- *youth → boys → boys setting up tent by lake → teenage boys pitching a tent at the lake in forest*

Few-Shot Synthetic Data Augmentation Our seed data, described in the previous section, has the correct format but is relatively small. To obtain sufficient data for knowledge distillation, we augment by generating synthetic caption hierarchies with the same format. In particular, we provide seed hierarchies as few-shot examples to LLM_{base} (see Section 2.1) and then generate a new line of text. As LLM_{base} is not instruction-tuned, no prompt text is needed. At each iteration, we provide ten randomly-selected examples of seed hierarchies as input to LLM_{base}, separated by newlines, and we proceed to generate text until another newline is predicted. We filter for outputs that have exactly four tiers. Note that we do not filter for logical entailment, as we find that the outputs sufficiently reflect the desired logical structure for the purposes of knowledge distillation (described in the next section) when used as-is.

While this process can be run indefinitely, we terminate this when we have generated $\sim 10K$ synthetic caption hierarchies. Examples of generated synthetic data include the following (displayed in the format $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4$):

- *music artist* $\rightarrow$ *singer* $\rightarrow$ *singer with group performs at event* $\rightarrow$ *pop singer with group performs at event*
- *textiles* $\rightarrow$ *towels* $\rightarrow$ *striped towels* $\rightarrow$ *a variety of striped towels in a range of colours and patterns - for sale on amazon*
- *flowers* $\rightarrow$ *poppies* $\rightarrow$ *fields of poppies.* $\rightarrow$ *a field of poppies, bright orange, on a sunny day*

Hierarchy Completion Distillation In order to learn to efficiently complete hierarchies from existing captions, we train a small language model $\mathcal{H}$ using the $\sim 10K$ synthetic hierarchies from Section 2.1 as supervision, thus distilling their contents into $\mathcal{H}$. We select the encoder-decoder model Flan-T5 [3], described further in Section 2.1, and define its input and target texts as follows:

Ground-truth hierarchies are represented as a single string $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4$ using a special token $\rightarrow$ to delimit tiers. At each step, we randomly select up to three initial tiers as the encoder prompt, and use the remaining tiers as targets. In addition, we randomly (with probability 0.5) reverse the directionality and use the special token $\leftarrow$ instead (i.e. $T_4 \leftarrow T_3 \leftarrow T_2 \leftarrow T_1$). Hence, possible input-target pairs include:

- Input: $T_1 \rightarrow$; Target: $T_2 \rightarrow T_3 \rightarrow T_4$
- Input: $T_1 \rightarrow T_2 \rightarrow$; Target: $T_3 \rightarrow T_4$
- Input: $T_4 \leftarrow T_3 \leftarrow$; Target: $T_2 \leftarrow T_1$

Note that this includes the case where no initial tiers are selected; we use the delimiter $\rightarrow$ or $\leftarrow$ as the input so the model has an indication as to the desired directionality of the output. In other words, these input-target pairs are:

- Input: $\rightarrow$; Target: $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4$
- Input: $\leftarrow$; Target: $T_4 \leftarrow T_3 \leftarrow T_2 \leftarrow T_1$

We train $\mathcal{H}$ on such inputs-target pairs using a language modelling objective. We train for three epochs on our synthetic hierarchies, using a batch size of 32, learning rate 1e-3, and the AdamW optimizer.$\mathcal{H}$ thus learns to complete hierarchies in either direction, and inference with $\mathcal{H}$ is faster and more reliable than the zero-shot LLM inference procedure described in Section 2.1.

Final Data Generation (*HierarCaps*) After training $\mathcal{H}$ to complete hierarchies, as described in the previous section, we apply it to captions from the Conceptual Captions [13] (CC) and MS-COCO [2, 8] datasets in order to put them in the context of full caption hierarchies for *HierarCaps*. We use captions from the CC train set to produce the *HierarCaps* train set, and captions from the COCO validation set to produce the *HierarCaps* test set, selecting captions with at least ten words.

We proceed as follows: Given a ground-truth caption $C = T_4$, we provide $T_4 \leftarrow$ as input to $\mathcal{H}$ and generate a batch of 64 output candidates. We filter these candidates for those of the format $T_3 \leftarrow T_2 \leftarrow T_1$ (i.e. exactly three output tiers), and then check if any of these candidates obey the following requirements: None of the T_i may be empty, no two adjacent tiers are identical (i.e. $T_1 \neq T_2 \neq T_3 \neq T_4$), T_3 must be at least two words shorter than T_4, T_3 must be at least ten characters shorter than T_4. Additional, for $i \in \{0, 1, 2\}$, we require that T_i be entailed by T_{i+1} while not entailing it (using our pretrained NLI model and following the procedure of Section 2.1). We output the first candidate obeying these conditions, if any satisfy them.

For *HierarCaps* test set items, we use the above procedure alone to generate hierarchies of positive texts (i.e. texts that are consistent with the ground-truth caption). For train set items, we additionally add negative examples (i.e. texts that contradict the ground-truth caption) as follows: Given the positive hierarchy $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4$, where T_4 is the original caption, we now apply $\mathcal{H}$ in the opposite direction to deliberately hallucinate negative texts. In particular, we provide the following prompts to $\mathcal{H}$:

- $\rightarrow$
- $T_1 \rightarrow$
- $T_1 \rightarrow T_2 \rightarrow$
- $T_1 \rightarrow T_3 \rightarrow T_3 \rightarrow$

We then generate text with $\mathcal{H}$ until the next $\rightarrow$ token, or until the end-of-string token is output (whichever comes first). $\mathcal{H}$ proceeds to generate outputs T_1', T_2', T_3', T_4' respectively. For each such T_i', we input the pair (T_i, T_i') into our pretrained NLI model (described in Section 2.1) and calculate $p_c^{(i)}$, the probability that these texts contradict each other; we filter to require that $p_c^{(ii')} > 0.5$. Additionally, if $i > 1$, we input the pairs (T_{i-1}, T_i') and (T_i', T_{i-1}) into the NLI model and check that T_{i-1} is entailed by T_i' but does not entail it, following the procedure of Section 2.1. As this procedure is failure-prone, we repeat it eight times (saving any valid T_i' values that are produced). If we finish with

T'_1, T'_2, T'_3, T'_4 values that obey the previous conditions, we output them as the desired negative examples.

This inference procedure yields $73K$ train set hierarchies, containing positive and negative texts, based on captions from CC. For our test set, after applying inference to captions from COCO to obtain positive hierarchies, we manually filter and correct a subset to obtain a hand-validated $1K$-item test set as a clean quantitative evaluation benchmark. While metrics are calculated using these manually-curated items alone, for qualitative results, we perform retrieval using the expanded set of candidate texts consisting of all automatic generations before hand-filtering, to yield more extensive predicted hierarchies. In our interactive visualization tool, we provide a visualization of all *HierarCaps* test set items along with a random subset of *HierarCaps* train set items.

```
[INST]
Shorten the following image caption. Feel free to remove details.
Put your answer in quotation marks: " "

"{}"
[/INST]

Sure! Here is the original caption and its shortened version:

Original caption: "{}"
Shortened version:
```

```
[INST]
You are given the following caption to an image:

"{}"

What one or two words should be filled in the blank in the following
alternative caption for the same image?

"A picture of (a/the) ___ ."

[/INST]

The best word or words to fill in the blank would be: "A picture of (a/the)
```

```
[INST]
What term describes a general concept that includes "{}"?
Answer in one or two words, and put your answer in quotation marks: " "
[/INST]

The best term that includes the given word or phrase is "
```

Fig. 1: Prompts used in the construction of *HierarCaps*. `{}` indicates the slot for inserting an input text. `[INST]` and `[/INST]` are special tokens used by the LLM Llama-2 to delineate the given instruction. Note that the prompts contain partial answer text by design, in order to enforce a uniform output format that may be easily parsed.

2.2 Entailment Cone Details

We provide further details related to the Entailment Cone (EC) framework of Ganea et al. [5] and its relation to our framework.

Partial Order Relations As discussed in the main paper, the EC framework defines a *partial order* on embeddings. We proceed to provide the mathematical definition of this term and its significance to entailment hierarchies.

A partial order on a set X is a binary relation $\leq$ defined on $X \times X$ which is reflexive ($x \leq x \ \forall \ x \in X$), antisymmetric ($a \leq b$ and $b \leq a$ imply $a = b$) and transitive ($a \leq b$ and $b \leq c$ imply $a \leq c$). Elements $a, b \in X$ are called *comparable* if either $a \leq b$ or $b \leq a$. X along with its associated partial order is called a partially ordered set or *poset*.

Partial orders have a natural application in the theory of logic, as logical entailment naturally forms a partial order over the the semantic denotations of assertions (i.e. over meanings) [14]. In particular, multimodal text-image data can be associated with a poset whose elements are the semantic denotations of images and texts[1], and whose partial order is identified with visual and textual entailment [16]. This motivates the design of the EC framework, which induces a partial order on embeddings corresponding to a logical entailment relation between items.

Derivation of RE Loss from Margin EC Loss We proceed to show that our RE loss function can be derived as the limit of a margin EC loss applied to contrastive pairs as the margin tends to infinity. See Section 3.5 for a comparison of results between our framework and training with an EC-based loss with various margins.

Consider the EC margin loss

$$\mathcal{L}_\alpha := \max(-\alpha, \pm(\Xi_{\mathbf{r}}(\mathbf{e}, \mathbf{e}') - \theta(d_{\mathbf{r}}(\mathbf{e})))),$$

where the sign ($\pm$) is 1 for positive entailment pairs and -1 for negative pairs (i.e. pairs $\mathbf{e}, \mathbf{e}'$ where $\mathbf{e}$ does not entail $\mathbf{e}'$). We find (in Section 3.5) that performance improves for larger margins, and the limiting case ($\alpha \to \infty$) is

$$\mathcal{L}_\infty := \pm(\Xi_{\mathbf{r}}(\mathbf{e}, \mathbf{e}') - \theta(d_{\mathbf{r}}(\mathbf{e}))).$$

We now use our key insight that the sum of such losses simplifies dramatically when for each positive pair $\mathbf{e}, \mathbf{e}'$ there is a negative pair $\mathbf{e}, \mathbf{e}''$. This matches the design of *HierarCaps*; for example, $\mathbf{e}, \mathbf{e}', \mathbf{e}''$ could be the embeddings of the texts $t =$*"animal"*, $t' =$*"goat"*, and $t'' =$*"portrait"* respectively, as t is entailed by t' and

[1] It is not a partial order over the images and texts themselves, but rather a *preorder*, as two distinct items can be logically equivalent (e.g. "this is a big cat" vs. "this is a large cat").

Model	#Param	#Param (text)
CLIP^{B}	151M	63M
CLIP^{L}	428M	123M
$\text{OpenCLIP}^{\text{B}}$	151M	63M
$\text{OpenCLIP}^{\text{H}}$	986M	353M
ALIGN	172M	109M

Table 1: Parameter counts of models used. Superscript letters indicate model size. #Param indicates total number of parameters, and #Param (text) indicates the number of parameters in the model's text encoder alone.

t contradicts t'' (and thus is not entailed by t''). Summing losses on these pairs gives

$$\begin{aligned}&\mathcal{L}_\infty(\mathbf{e},\mathbf{e}') + \mathcal{L}_\infty(\mathbf{e},\mathbf{e}'')\\ &= \Xi_{\mathbf{r}}(\mathbf{e},\mathbf{e}') - \theta(d_{\mathbf{r}}(\mathbf{e})) - \Xi_{\mathbf{r}}(\mathbf{e},\mathbf{e}'') + \theta(d_{\mathbf{r}}(\mathbf{e}))\\ &= \Xi_{\mathbf{r}}(\mathbf{e},\mathbf{e}') - \Xi_{\mathbf{r}}(\mathbf{e},\mathbf{e}''),\end{aligned}$$

which motivates the definition of the RE loss as

$$\mathcal{L}_{RE} := \Xi_{\mathbf{r}}(\mathbf{e},\mathbf{e}') - \Xi_{\mathbf{r}}(\mathbf{e},\mathbf{e}'').$$

Note that this loss does not depend on the half-aperture function θ.

2.3 Model Details

For our results on hierarchical understanding tasks, we use the following model checkpoints (all from Hugging Face model hub):

- `openai/clip-vit-base-patch32`
- `openai/clip-vit-large-patch14`
- `laion/CLIP-ViT-B-32-laion2B-s34B-b79K`
- `laion/CLIP-ViT-H-14-laion2B-s32B-b79K`
- `kakaobrain/align-base`

These correspond respectively to the models CLIP-Base, CLIP-Large, OpenCLIP-Base, OpenCLIP-Huge, and ALIGN respectively. See Table 1 for parameter counts of these models.

2.4 Training Details

When performing model fine-tuning, we train for a single epoch on *HierarCaps*, using batch size 8 and the AdamW optimizer with learning rate 1e-7. For each model under consideration, we only train its text encoder while keeping all other weights frozen.

The total loss used for training is $\mathcal{L}_{total} = \lambda_{RE}\tilde{\mathcal{L}}_{RE} + \lambda_{reg}\mathcal{L}_{reg}$, as described in the main paper. We use coefficients $\lambda_{RE} = 1.0$ and $\lambda_{reg} = 10.0$ (ablated in Section 3.1).

The aggregate RE loss $\tilde{\mathcal{L}}_{RE}$ is calculated as follows: Each item in the *HierarCaps* train set contains a positive caption hierarchy (P_1, P_2, P_3, P_4) with accompanying negative captions (N_1, N_2, N_3, N_4). As described in the main paper, for each $i \in \{1, 2, 3\}$ the caption triplet (P_i, P_{i+1}, N_i) can be used to calculate a exterior angle-based contrastive loss. For tier i in item j of a minibatch, denote this loss by $\mathcal{L}_{RE}^{(i,j)} = \Xi_{\mathbf{r}}(\mathbf{e}, \mathbf{e}') - \Xi_{\mathbf{r}}(\mathbf{e}, \mathbf{e}'')$, where $(\mathbf{e}, \mathbf{e}', \mathbf{e}'')$ are the embeddings of the respective elements of the corresponding caption triplet. We take the mean value of this loss across such i and j to yield mean RE loss $\mathcal{L}_{RE}^{mean}$. In addition, we perform hard example mining as follows: Let (i', j') denote the (i, j) values that maximize $\Xi_{\mathbf{r}}(\mathbf{e}, \mathbf{e}')$, and let (i'', j'') denote the (i, j) values that minimize $\Xi_{\mathbf{r}}(\mathbf{e}, \mathbf{e}'')$. In other words, these are the indices of items with maximal terms in the RE loss function. We then define the mini-max loss $\mathcal{L}_{RE}^{mm} := \mathcal{L}_{RE}^{(i',j')} + \mathcal{L}_{RE}^{(i'',j'')}$. Finally we define the aggregate loss $\tilde{\mathcal{L}}_{RE} := \mathcal{L}_{RE}^{mean} + \mathcal{L}_{RE}^{mm}$. This loss both encourages the RE loss to decrease across all items (first term) as well as focusing on the hardest examples (second term).

2.5 *HierarCaps* Inference Details

For qualitative and quantitative results on the *HierarCaps* test set, we perform hierarchical image→text retrieval. Following the methodology of Desai et al. [4], we select 50 equally-spaced points between the root node and closest text embedding to the given image; at each point, we retrieve the closest candidate text within the given radius, as measured by embedding cosine similarity. As the closest points to the origin often lead to a degenerate solution where only a single text embedding exists within the given radius, we omit the closest prediction to the origin.

Quantitative metrics are calculated with respect to the manually-curated test set of 1K ground-truth caption hierarchies. For qualitative results we use an expanded candidate set of all automatic generations before hand-filtering, to yield more extensive predicted hierarchies. These consist of text produced using automatic hierarchy completion (see Section 2.1) on ground-truth reference captions corresponding to each image in the COCO validation set. We release both the manually-curated test set used for metric calculations as well as the expanded set used for qualitative results.

2.6 External Benchmark Experimental Details

For tests on HyperLex, we evaluate by inserting lexical items (e.g. *girl*) into the prompt *a picture of a* ____ (e.g. *a picture of a girl*) before embedding them, as we observe that this empirically outperforms embedding the lexical items as-is on the HyperLex evaluation metrics. We note the similarity to the common practice of inserting class names into similar prompts when using CLIP for zero-shot classification [12].

	HierarCaps			HyperLex		BREEDS		COCO	
Model	P	R	τ_d	ρ_{all}	ρ_N	R@1	R@5	R@1	R@5
CLIP^{B}	0.14	0.36	0.89	0.44	0.48	0.22	0.50	0.30	0.55
$\text{CLIP}^{\text{B}}_{\text{FT}}$	0.15	0.47	0.99	0.51	0.55	0.24	0.55	0.31	0.56
fixed root ($\bar{\mathbf{e}}$)	0.16	0.44	0.98	0.50	0.55	0.24	0.54	0.32	0.56
fixed root ($\mathbf{0}$)	0.21	0.19	0.54	0.23	0.23	0.04	0.09	0.31	0.55
fixed root ($\mathbf{e}_\emptyset$)	0.14	0.47	0.98	0.50	0.55	0.24	0.54	0.31	0.55
only learning root	0.15	0.37	0.93	0.47	0.51	0.24	0.53	0.29	0.53
$\lambda_{reg} = 0$	0.15	0.45	0.99	0.49	0.53	0.21	0.52	0.26	0.49
3 tiers	0.17	0.50	0.93	0.40	0.44	0.17	0.41	0.30	0.54
no negative examples	0.16	0.42	0.99	0.48	0.52	0.25	0.54	0.28	0.52
FT with CLIP objective	0.14	0.28	0.95	0.43	0.47	0.22	0.49	0.32	0.58
training from scratch	0.00	0.01	0.93	0.11	0.12	0.00	0.00	0.00	0.00

Table 2: Ablation study results. We first reproduce the results on our suite of tasks for a CLIP model before and after model alignment (fine-tuning). Below this, we display fine-tuning results when removing key components of our framework. Task names and metrics follow the notation of the results table in the main paper.

For tests on BREEDS, we reiterate that we predict label pairs (x, y) for an image to maximize the score $c_x + c_y + C \cdot \Xi_{\mathbf{r}}(x, y)$, where c_x and c_y are embedding similarities between each label and the image, C is a constant chosen by cross-validation, and $\Xi_{\mathbf{r}}(x, y)$ is the exterior angle between embeddings of texts x and y. Label embeddings are produced using the prompt ensemble of Radford et al. [12]. We select the constant C as follows: We split test items into two equally-sized folds, F_0 and F_1. For each metric (R@1, R@5), we find the value of C among 20 equally-spaced values between 0 and 0.2 which maximizes the metric on F_i and output its value v_{i-1} on F_{1-i}. Finally we report the average value $(v_0 + v_1)/2$.

3 Additional Results and Comparisons

3.1 Ablation Study

We show the contribution of core elements of our fine-tuning framework by ablating each in turn. See Table 2 for quantitative results of these ablations, which we describe below, recalling that the first three benchmarks listed (*HierarCaps*, HyperLex, BREEDS) measure hierarchical knowledge while the last (COCO) measures standard multimodal understanding (i.e. pretraining knowledge).

We ablate the effect of a learnable root corresponding to the embedding of the empty string ($\mathbf{e}_\emptyset$) by comparing it to the methodology of Desai et al. [4], who instead use the empirical mean of the embedding vector of train set items as the root position for CLIP. We use $\bar{\mathbf{e}}$ to indicate this empirical mean, calculating it as the (unit-normalized) mean of the embeddings of all texts in the *HierarCaps*

train set (treating each caption from each tier, both positive and negative, as a separate text to embed). We do not use the corresponding images in this calculation since all our fine-tuning procedure uses only the textual data from *HierarCaps*. As is seen in the table, this leads to a degradation in hierarchical performance as reflected in poorer *HierarCaps* recall relative to our original fine-tuning method. We also consider two additional alternatives: Using the origin ($\mathbf{0}$) as a fixed root, and by using the original embedding $\mathbf{e}_\emptyset$ of the empty string as a fixed root vector (i.e. not allowing its position to change during fine-tuning). As is seen in the table, the initialization of the root has a critical effect, as the model fails by most hierarchical understanding metrics when the root is fixed at $\mathbf{0}$, while when the root is instead fixed at the original embedding $\mathbf{e}_\emptyset$, performance is much closer to our original fine-tuning procedure.

Conversely, we also perform an ablation in which we learn the root embedding position alone while keeping all other embeddings fixed ("only learning root" in Table 2). We see that this yields performance close to the base CLIP$^{\text{B}}$ model without fine-tuning, demonstrating that the root embedding position alone is insufficient for our fine-tuning framework.

We also ablate the effect of our regularization loss by fine-tuning with $\lambda_{reg} = 0$ (see Section 2.4). As is seen in the table, this loss is needed to avoid a drop in performance on a standard multimodal task (COCO text→image retrieval). In other words, the regularization loss prevents catastrophic forgetting of pretraining knowledge.

To show the impact of the four-tiered structure of *HierarCaps*, we ablate by removing an intermediate tier (the second-most generic items) for fine-tuning. Results are listed as "3 tiers" in Table 2. We see a particularly significant drop in performance on metrics such as τ_d for *HierarCaps* and performance on HyperLex, which are heavily weighted towards understanding lexical entailment and the relative semantics of generic items. This matches the observation that VLMs are known to struggle with more generic concepts, as discussed in the main paper, and hence the more generic tiers provide particularly important supervision during our fine-tuning process.

We ablate the effect of the positive-negative example pairs used during fine-tuning. We ablate the use of negative examples by setting all loss terms corresponding to negative pairs (i.e. $\Xi_{\mathbf{r}}(\mathbf{e}, \mathbf{e}'')$, where $(\mathbf{e}, \mathbf{e}', \mathbf{e}'')$ is a caption triplet as described in the main paper) to zero during training. As is seen in the table, this degrades the quality of representations across tasks, particularly hierarchical understanding tasks.

Finally, we test two alternatives to our fine-tuning method. We test using the original CLIP contrastive objective (instead of our RE loss) using text labels randomly sampled from from each hierarchy. This mostly performs similarly to pretrained CLIP, far under-performing our RE fine-tuning across tasks and thus suggesting that our results cannot be attributed only to learning the general semantics of the *HierarCaps* dataset. In addition, we test training from scratch with our loss rather than using pretrained CLIP weights. We initialize the CLIP-base architecture randomly (and remove the regularization loss minimizing de-

viation from the initial weights). As our training only adjusts the text encoder, this exhibits random performance on image matching tasks (*HierarCaps* P & R, BREEDS, COCO). On text-only tasks we find that it has learned some hierarchical understanding, while mostly underperforming pretrained CLIP. This strengthens our core motivation of probing and optimizing the emergent geometry of pre-trained VLMs, which evidently have already learned embedding configurations which are highly compatible with visual-semantic hierarchies.

3.2 Effect of Text Length

We note that text length may be partially correlated with specificity or generality, as seen in Figure 2 with more specific tiers of *HierarCaps* generally being longer. To test whether our RE fine-tuning results are mainly attributable to text length, we examine the performance of $\text{CLIP}^{\text{B}}_{\text{FT}}$ on *HierarCaps* when controlling for text length. We do this by comparing our overall results to those when restricting to controlled subsets of the *HierarCaps* test set – when either the first two (most general) tiers have the same word count (72% of the test data), or when the last two (most specific) tiers differ by at most four words (34% of the test data). In both cases, we find all metrics (P, R, τ_d) to differ by less than 0.015 in value with their values on the entire test set. We also note that the external HyperLex dataset consists of single words alone, and our results show a significant improvement on this benchmark after fine-tuning. Overall, we conclude that, while text length may serve as a cue to specificity, the observed hierarchical effects in the VLMs we investigate are not primarily due to text length.

3.3 Additional Cross-Modal Task Results

Table 3 shows performance on additional standard cross-modal tasks and metrics, along with those reported in our main paper. These include text→image retrieval on MS-COCO [2,8] (val set) and Flickr30K [20] (test set), and zero-shot image classification on CIFAR-10 and CIFAR-100 [6]. For classification tasks, we use the prompts from Radford et al. [12] along with their procedure for aggregating scores across prompts. Results are shown in Table 3. As is seen there, our fine-tuning procedure has a negligible impact on performance across these tasks (while boosting hierarchical understanding, as described in the main paper).

3.4 Additional Qualitative Results

Additional qualitative results can be viewed using our interactive visualization tool; see Section 1.

Browsing the uncurated *HierarCaps* train set illustrates the automatically-produced caption hierarchies in this benchmark, which we manually curated in the test set. This was necessary due to various types of noise which could occur as a result of the automatic generation process; we illustrate representative

	COCO			Flickr30K			CIFAR-10	CIFAR-100
Model	R@1	R@5	R@10	R@1	R@5	R@10	acc	acc
CLIP^{B}	0.30	0.55	0.66	0.59	0.83	0.90	0.90	0.66
$\text{CLIP}^{\text{B}}_{\text{FT}}$	0.31	0.56	0.67	0.60	0.84	0.91	0.90	0.65
CLIP^{L}	0.36	0.60	0.70	0.65	0.87	0.92	0.96	0.78
$\text{CLIP}^{\text{L}}_{\text{FT}}$	0.36	0.61	0.71	0.66	0.88	0.93	0.96	0.79
$\text{OpenCLIP}^{\text{B}}$	0.39	0.65	0.75	0.67	0.88	0.93	0.94	0.76
$\text{OpenCLIP}^{\text{B}}_{\text{FT}}$	0.39	0.65	0.74	0.67	0.88	0.93	0.93	0.77
$\text{OpenCLIP}^{\text{H}}$	0.48	0.73	0.81	0.78	0.94	0.97	0.98	0.86
$\text{OpenCLIP}^{\text{H}}_{\text{FT}}$	0.48	0.73	0.81	0.77	0.94	0.97	0.98	0.86
ALIGN	0.42	0.67	0.77	0.74	0.92	0.96	0.78	0.53
ALIGN_{FT}	0.42	0.67	0.77	0.74	0.92	0.96	0.78	0.53

Table 3: Results on standard cross-modal tasks before and after fine-tuning. Superscript letters indicate model size. *acc* indicates categorical accuracy, and R@k refers to recall at k. As seen above, our fine-tuning has a negligible impact on performance on these tasks.

examples of this in Figure 2. As is seen there, such noise may include include minor misunderstanding of wording on the part of the language model (image 1), the use of abstract terms which are not incorrect per se but do not concretely describe the image (images 2–3), and noise inherited from the long image captions in the last tier which come from an existing captioning dataset (image 4). These are often attributable to the limited capacity of the language model used to produce these captions and the production of caption hierarchies with a text-only pipeline, both drawbacks of the efficient use of the small distilled text-only language model $\mathcal{H}$.

Fig. 2: Automatically-generated hierarchies from the HierarCaps train set, with inaccuracies highlighted. We show similar synthetic images rather than the original images due to licensing.

	HierarCaps		
Model	P	R	τ_d
CLIP^{B}	0.14	0.36	0.89
$\text{CLIP}^{\text{B}}_{\text{FT}}$	0.15	0.47	0.99
$\alpha = 0$	0.13	0.30	0.97
$\alpha = \pi/8$	0.14	0.33	0.97
$\alpha = \pi/4$	0.14	0.35	0.98
$\alpha = \pi/2$	0.17	0.46	0.99

Table 4: Results when fine-tuning with EC-based losses. We first reproduce our results for a CLIP model before and after fine-tuning using our full framework including a RE loss. We then report results when substituting an EC loss for our contrastive RE loss, as described in Section 3.5. In particular, we see increased performance as the margin α increases, which we show (in Section 2.2) has the RE loss as its limiting case.

3.5 Comparison to Entailment Cones

We compare our RE framework to the Entailment Cone (EC) framework of Ganea et al. [5] by replacing the loss $\mathcal{L}_{RE}$ in fine-tuning with an EC loss. In particular, we consider margin EC losses of the form

$$\mathcal{L}_{EC}^{(\alpha,\epsilon)} := [\pm(\Xi_{\mathbf{r}}(\mathbf{e}, \mathbf{e}') - \theta_\epsilon + \alpha)]^+$$

where $[x]^+ = \max(0, x)$ is the positive part (ReLU) function, α is the chosen margin, and $\pm$ is positive for positive pairs and negative for negative pairs. The entailment cone half-angle θ_ϵ is defined by $\sin\theta_\epsilon = \min(1, \epsilon/d_{\mathbf{r}}(\mathbf{e}))$. We use this loss by replacing $\Xi_{\mathbf{r}}(\mathbf{e}, \mathbf{e}')$ and $\Xi_{\mathbf{r}}(\mathbf{e}, \mathbf{e}'')$ in the calculation of $\mathcal{L}_{RE}$ with the expression above, using fixed $\epsilon = 0.05$, and proceeding to aggregate losses across tiers and samples as described in Section 2.4. See Table 4 for results using various values of α, beyond those reported in the main paper. From these results we see improved performance as the margin α increases, motivating the derivation of the RE loss from this loss as $\alpha \to \infty$, as described in Section 2.2.

3.6 Comparison to Existing HyperLex Methods

See Table 5 for a comparison of our results on HyperLex (using a CLIP model as-is as well as after our fine-tuning), along with various dedicated methods for learning lexical entailment. We reproduce the ρ_{all} and ρ_N metrics reported by each work – the Spearman correlation between predicted scores and ground-truth entailment scores from HyperLex, for all items and for nouns respectively. These results show that our method is competitive in this field; while we do not surpass the state-of-the-art for dedicated methods, we do show that CLIP can achieve performance of a similar order of magnitude to existing methods, either when used zero-shot or with model alignment (fine-tuning). We also emphasize the significant effect of model alignment, which improves both ρ_{all} and ρ_N by

Model	ρ_{all}	ρ_N
Ours		
CLIP^{B}	0.44	0.48
$\text{CLIP}^{\text{B}}_{\text{FT}}$	0.51	0.55
Dedicated		
LEAR [18]	0.69	0.71
DOA-E [1]	-	0.59
HyperVec [10]	0.54	-
Poincaré [11]	-	0.51
PARA [9, 10]	0.32	-
FR [17]	0.28	0.28

Table 5: Results on the HyperLex lexical entailment benchmark, comparing our results when probing CLIP (before or after model alignment) to existing dedicated methods. We use the following abbreviations: Frequency Ratio (FR), PARAGRAM (PARA), Order Embeddings (OE). We reproduce values from the cited works, noting that some only report one of ρ_{all} or ρ_N. We see that our method is competitive with existing dedicated methods for this task.

0.07 and reduces the gap between CLIP and the highest-performing dedicated models.

3.7 BREEDS Single Prompt Baseline Results

To provide a baseline comparison and provide justification for our inference method, we evaluate performance on BREEDS when predicting label pairs inserted into a single prompt. We use the same BREEDS subsets and labels as discussed in the main paper. To predict a pair of labels (x, y), we insert both labels into a prompt with two slots (e.g. `a {} of type {}`, where `{}` indicates a slot for inserting text), and yield the pair of distinct labels (x, y) which maximizes the embedding similarity of the given prompt with the image. In Table 6, we report performance for various such prompts on a CLIP model evaluated both before and after model alignment; we also reproduce the performance of our probing method from the main paper. As seen in the table, zero-shot classification of paired hierarchical labels using this single prompt-based inference method yields near-random performance, unlike our proposed inference method.

References

1. Athiwaratkun, B., Wilson, A.G.: Hierarchical density order embeddings. arXiv preprint arXiv:1804.09843 (2018)
2. Chen, X., Fang, H., Lin, T.Y., Vedantam, R., Gupta, S., Dollár, P., Zitnick, C.L.: Microsoft coco captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325 (2015)

	CLIP^{B}		$\text{CLIP}^{\text{B}}_{\text{FT}}$	
Prompt	R@1	R@5	R@1	R@5
`a picture of a {}, a type of {}`	0.00	0.02	0.00	0.02
`a {} of type {}`	0.01	0.04	0.01	0.03
`this is a {}, a type of {}`	0.01	0.03	0.00	0.03
`a picture of a {} of type {}`	0.01	0.04	0.01	0.03
`a picture of a {} which is a {}`	0.00	0.03	0.00	0.02
(ours)	0.22	0.50	0.24	0.55

Table 6: Baseline results for classification of BREEDS label pairs using a single prompt containing two slots for labels. In the above prompts, `{}` indicates the slot for inserting a label text. We also reproduce results for our proposed inference method from the main paper at the bottom. We see that the baseline yields near-random performance and is significantly outperformed by our proposed method.

3. Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al.: Scaling instruction-finetuned language models. arXiv preprint arXiv:2210.11416 (2022)
4. Desai, K., Nickel, M., Rajpurohit, T., Johnson, J., Vedantam, S.R.: Hyperbolic image-text representations. In: International Conference on Machine Learning. pp. 7694–7731. PMLR (2023)
5. Ganea, O., Bécigneul, G., Hofmann, T.: Hyperbolic entailment cones for learning hierarchical embeddings. In: International Conference on Machine Learning. pp. 1646–1655. PMLR (2018)
6. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
7. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461 (2019)
8. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. pp. 740–755. Springer (2014)
9. Mrkšić, N., Séaghdha, D.O., Thomson, B., Gašić, M., Rojas-Barahona, L., Su, P.H., Vandyke, D., Wen, T.H., Young, S.: Counter-fitting word vectors to linguistic constraints. arXiv preprint arXiv:1603.00892 (2016)
10. Nguyen, K.A., Köper, M., Walde, S.S.i., Vu, N.T.: Hierarchical embeddings for hypernymy detection and directionality. arXiv preprint arXiv:1707.07273 (2017)
11. Nickel, M., Kiela, D.: Poincaré embeddings for learning hierarchical representations. Advances in neural information processing systems **30** (2017)
12. 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: International conference on machine learning. pp. 8748–8763. PMLR (2021)
13. Sharma, P., Ding, N., Goodman, S., Soricut, R.: Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In: Proceed-

ings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 2556–2565 (2018)
14. Toledo, A., Alexandropoulou, S., Katrenko, S., Klockmann, H., Kokke, P., Winter, Y.: Semantic annotation of textual entailment. In: Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers. pp. 240–251 (2013)
15. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C.C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P.S., Lachaux, M.A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E.M., Subramanian, R., Tan, X.E., Tang, B., Taylor, R., Williams, A., Kuan, J.X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., Scialom, T.: Llama 2: Open foundation and fine-tuned chat models (2023)
16. Vendrov, I., Kiros, R., Fidler, S., Urtasun, R.: Order-embeddings of images and language. In: Bengio, Y., LeCun, Y. (eds.) ICLR (2016)
17. Vulić, I., Gerz, D., Kiela, D., Hill, F., Korhonen, A.: Hyperlex: A large-scale evaluation of graded lexical entailment. Computational Linguistics **43**(4), 781–835 (2017)
18. Vulić, I., Mrkšić, N.: Specialising word vectors for lexical entailment. arXiv preprint arXiv:1710.06371 (2017)
19. Williams, A., Nangia, N., Bowman, S.: A broad-coverage challenge corpus for sentence understanding through inference. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 1112–1122. Association for Computational Linguistics (2018), `http://aclweb.org/anthology/N18-1101`
20. Young, P., Lai, A., Hodosh, M., Hockenmaier, J.: From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. Transactions of the Association for Computational Linguistics **2**, 67–78 (2014)