Concept Space Prior in Language and Vision models

Sungjun Han
University of Stuttgart
st175409@stud.uni-stuttgart.de

Esra Dönmez
University of Stuttgart
st172173@stud.uni-stuttgart.de

Abstract

Problems at the intersection of language and vision are of significant importance, both as challenging research questions as well as industrial applications. Models based on transfer learning from a model pretrained using a large amount of unannotated multi-modal data is one of the most successful approach in the field of language and vision in recent times. Another approach tangent with the pretrain-transfer approach that focuses on using the appropriate structural priors without any pretraining data has also shown to be competitive. One model of this family called Neural State Machine used a prior that we refer to as the concept space prior to map both modalities to the same abstract semantic space for reasoning. This prior draws inspiration from the way humans think. In this work, we study the importance of the concept space prior in language and vision models. We do this by equipping a similar model called Language-conditioned Graph Network with the prior. We propose three different ways of modifying the model: Local-VCS, Dynamic-VCS, and MCS. We show that the priored LCGN is able to do as well as the original model and examine the results between the proposed models to elucidate the necessary mechanisms in mapping the modalities to the abstract concept space that are needed to support the prior.

1 Introduction

Grounded understanding refers to the ability to connect abstract concepts and natural language to their extensions in the real world. The importance of grounded understanding in building intelligent machines has received renewed attention in the recent decade through the development of the field of Language and Vision (LV) (Bisk et al., 2020; Lu et al., 2019). LV tasks require visual understanding by responding to natural language in the context of visual stimuli. Popular tasks include image-captioning, visual question answering, and referring expression matching.

The state-of-the-art approaches in LV are transformer (Vaswani et al., 2017) based models (Lu et al., 2019; Chen et al., 2020; Tan and Bansal, 2019) pretrained on a massive amount of weakly aligned visual-linguistic data collected from the web, such as Conceptual Captions (Sharma et al., 2018). These models represent the image as a bag of object features collected from an object-detection model and the question as a sequence of words. Then, these models rely on the transformer's self-attention mechanism to construct joint-visiolinguistic representations from the input. Hence, these models can be understood as a general graphical network where the nodes consist of the detected objects from the image and the words from the question. These approaches use little to no additional structural priors to guide the representation learning.

Another graphical approach (Hu et al., 2019; Hudson and Manning, 2019b) has also been popular that learn to reason on the graph consisting of only the detected objects by conditioning on the information extracted from the question. These models are not pretrained as they are usually too specific to be used with the popular self-supervising objectives employed by the transformer models. Despite this, one of the models named Neural State Machine (NSM) (Hudson and Manning, 2019b) was able to perform better than some of the pretrained transformer models such as LXMERT (Tan and Bansal, 2019) on the challenging GQA dataset (Hudson and Manning, 2019a).

NSM employed a special structural prior to map both modalities to the abstract concept space defined by a concept vocabulary in constructing and reasoning over the graph.

We refer to NSM's structural prior as the **concept space prior**. This prior draws inspiration from the way humans are believed to think. Humans reason by manipulating a relatively small number of high-level concepts where a concept is represented by a prototype or a set of typical exemplars (Murphy, 2002; Bealer, 1998; Kahneman, 2011). Another work (Bengio, 2019) even claimed that such an ability to construct and operate over a small discrete set of high-level concepts is the key to consciousness. The core idea behind this prior is the simplification of the reasoning process by operating on a discrete set of concepts. For example, in visual question answering, since both modalities are mapped to the same semantic space, the reasoning process simply involves computing the similarities between the identified objects from the image and the question to identify relevant objects for classification. This prevents the model from learning spurious correlations from the feature vectors and the answers leading to better generalization.

The ablation study from the NSM's authors suggested that the concept space prior was crucial in the success of the model. However, it is difficult to assess the importance of the prior in LV models *in general* as the conclusion made from an ablation study on the necessity of a core assumption is only valid in the presence of other necessary assumptions of the model.

In this work, we investigate the importance of the NSM's concept space prior in LV models[1]. We do this by equipping a similar graph model called Language-conditioned Graph Network (LCGN) (Hu et al., 2019) with the same prior by defining a concept vocabulary. We believe that LCGN is a good testbed for studying this question as it works on a similarly constructed graph from the detected objects as NSM, but using a more general mechanism to reason over the graph. Hence, it would be possible to deduce the importance of the concept space prior separate from the other assumptions of NSM and find any underlying architectural decisions that are necessary to support the prior.

We introduce different ways of introducing the concept space prior to LCGN, namely *Local-Vector-Concept-Space*(Local-VCS), *Dynamic-Vector-Concept-Space*(Dynamic-VCS), and *Matrix-Concept-Space*(MCS) and report on their results on a visual question answering task. Each model implements the concept space prior with a different architecture. The VCS models map the objects to the concept space by using only the prototypes that represents the concepts. In Local-VCS, the mapping operation is done only once in the beginning while for Dynamic-VCS, it is done multiple times dynamically after each message passing step of the graph. MCS maps the objects by considering their relevance to each concept independently of the other concepts and represents the objects using the typical exemplars from the relevant concepts. Similar to Local-VCS, it maps only once in the beginning. In the next sections, we will describe in detail how these models work. By comparing the results between these models, we show that the concept space prior requires a specific architectural mechanism implemented in MCS to support it. We further analyse the models by introducing three additional hyperparameters for the models: *stem function*, *use of concept vocabulary*, and *fixing concept vectors*. The results from varying the stem function further support the aforementioned conclusion. The results from training the model with or without a constructed concept vocabulary add to the validity of the above conclusion by showing that the three proposed models all rely on the concept space prior for reasoning. Finally, the results from allowing the model to modify the concept space during training show that it is a difficult to do so from the supervision signal alone.

2 Models

In this section, we briefly describe the architectures of NSM and LCGN. High level descriptions of the two models are given in Figure 1 and Figure 2. For more details, we refer the readers to their respective original papers. Both models work on the graph $G = (V, E)$ where $V = \{o_i\}_{i=1}^{N}$. o_i's are usually features from an object-detection model such as Faster-RCNN (Ren et al., 2015). LCGN assumes the existence of edges between all nodes $E = \{e | \forall o_i, o_j \in V, e = (o_i, o_j)\}$ while NSM relies on a scene-graph generation algorithm to identify the edges between the objects. Both models

[1]We release the code for our models here: https://github.tik.uni-stuttgart.de/PascalTilli/AML-group-3

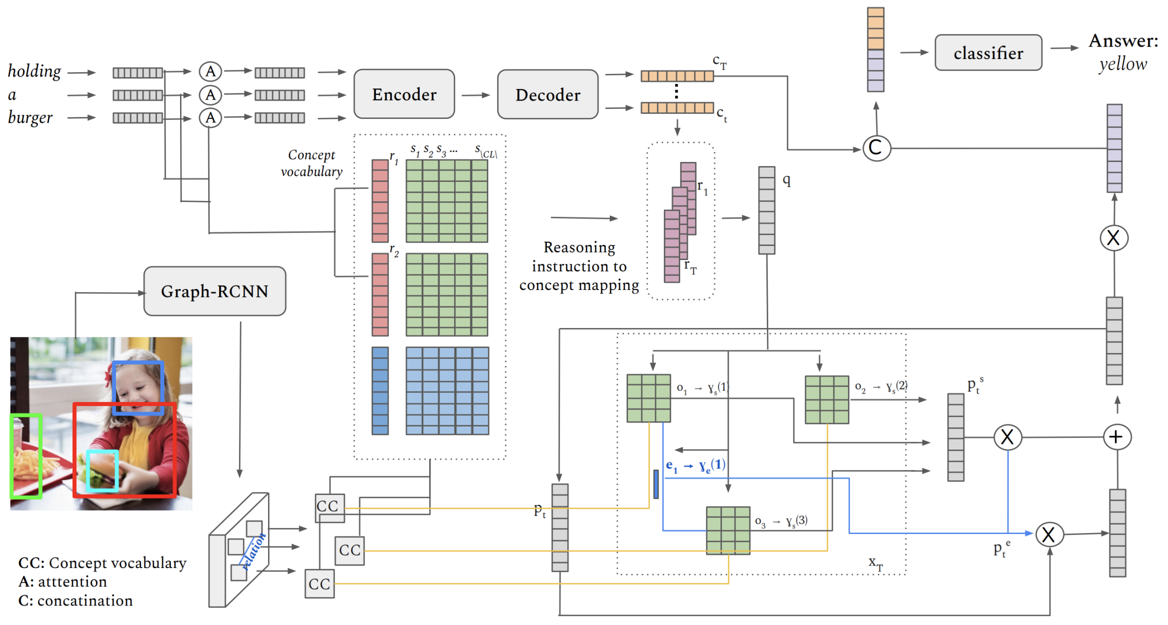

Figure 1: Model architecture of NSM.

use an encoder to produce a sequence of linguistic commands of length $T : c = [\mathbf{c}_1, ..., \mathbf{c}_T], \mathbf{c}_t \in \mathbb{R}^l$. For each time step t, the models use $\mathbf{c}_t$ to condition the communication between the nodes in G.

2.1 NSM

2.1.1 Concept Space Prior

NSM employs the concept space prior which constrains the model to operate over a discrete set of concepts to represent the semantics of an input image and a question. In order to structure the concept space, they define a concept vocabulary C of size $L + 2$. The first concept C_0 holds the object identity. The next L concepts $\{C_i\}_{i=1}^{L}$ hold various attribute concepts such as colors or materials. Finally, the last concept C_{L+1} holds the visual relations between objects. Each concept C_i is defined in terms of a variable number of concept-members $\mathbf{s}_j$. For example, the color concept would have members such as ”red” or ”green”. Then, the concept vocabulary is initialized by assigning a d-dimensional feature vector to each concept-member $C_l = \{\mathbf{s}_j\}_{j=1}^{|C_l|}, \mathbf{s}_j \in \mathbb{R}^d$ using GloVe embeddings (Pennington et al., 2014). Each concept is given a feature vector $\mathbf{r}_i \in \mathbb{R}^d$ by averaging the feature vectors of its members. Hence, $\mathbf{r}_i$ represents the prototype that best resembles its category members.

2.1.2 Mapping to Concept Space

NSM uses a scene-graph generation algorithm based on Graph-RCNN by (Yang et al., 2018) to generate the graph $G = (V, E)$. Graph-RCNN was modified to additionally classify each object in L ways for the attribute concepts $\{C_i\}_{i=1}^{L}$. Then each object o is mapped to the concept space by a set of vectors $\{\mathbf{o}^j\}_{j=0}^{L}$matrix by using the outputted probability distributions to do weighted-averaging of the feature vectors from C for each concept. Similarly, the edges are represented as d-dimensional vectors using the concept vocabulary.

The question, represented as a sequence of tokens, is converted into a sequence of feature vectors $\mathbf{w}_i$ using the Glove embeddings. Then in order to map the question to the concept space, each feature vector is mapped to the closest concept by attending to the concept embeddings defined in C. These mapped vectors then become the inputs to an LSTM-based encoder-decoder (Sutskever et al., 2014) architecture. The decoder is then rolled out for a fixed number of steps T, resulting in T reasoning instructions $\{\mathbf{c}_t\}_{t=1}^{T}$:

$$\mathbf{c}_t = softmax(\mathbf{h}_t \mathbf{V}^T)\mathbf{V} \tag{1}$$

where $\mathbf{h}_t$ is a hidden state of the decoder at time step t and $\mathbf{V} \in \mathbb{R}^{|C|\times l}$. Essentially, at each step, the decoder hidden state is classified into one of $|C|$ concepts to form a reasoning instruction.

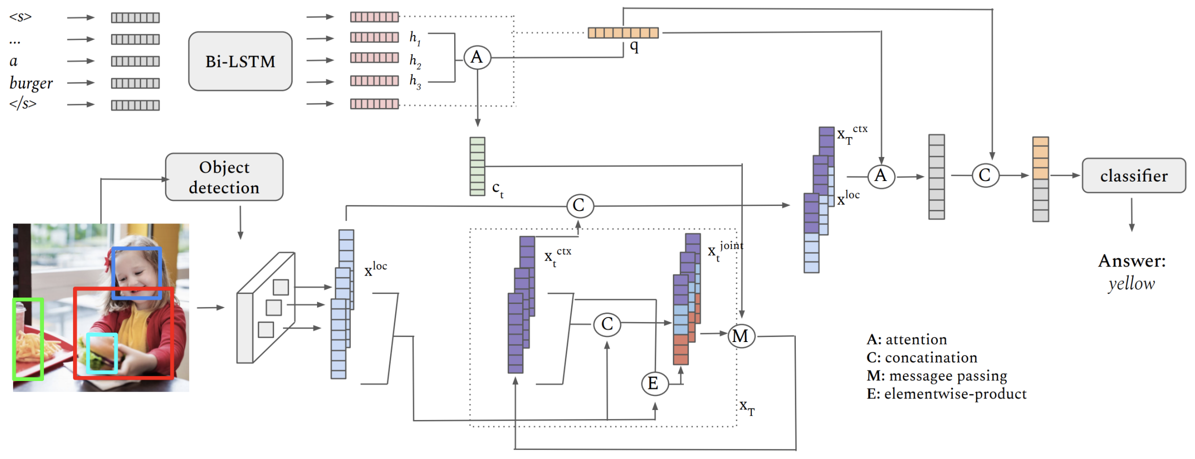

Figure 2: Model architecture of LCGN.

2.1.3 Reasoning over Graph

With the graph and the reasoning instructions mapped to the concept space, NSM iteratively calculates the probability mass to assign to each node (i.e. the object). First, the reasoning instruction $\mathbf{c}_t$ is used to further classify which concept it is talking about by comparing itself with the prototype concept vectors $\mathbf{r}_i$'s.

$$\mathbf{q}_t(j) = softmax(\mathbf{r}_j \mathbf{W} \mathbf{c}_t)_j \tag{2}$$

The probability $\mathbf{q}_t(j)$ represents how much information from the $j-$th concept object feature vectors will be considered at time step t. Second, this is used to weigh how much score to assign to each object $\mathbf{F}$ and each edge in terms of their relevance to the reasoning instruction:

$$\gamma_t(o) = \sigma(\sum_{j=0}^{L} \mathbf{q}_t(j)(\mathbf{c}_t \odot \mathbf{W}_j o^j)) \tag{3}$$

$$\gamma_t(e) = \sigma(\mathbf{q}_t(L+1) \odot \mathbf{W}_{L+1} e) \tag{4}$$

The scores are converted to probabilities by taking a linear combination of the scores at the current time step and the probabilities from the previous time step according to the edge scores.

$$\mathbf{p}^s_{t+1} = softmax_{o \in V}(\mathbf{W}_s \cdot \gamma_t(o)) \tag{5}$$

$$\mathbf{p}^r_{t+1} = softmax_{s \in S}(\mathbf{W}_r \cdot \sum_{(\mathbf{o}', \mathbf{o}) \in E} \mathbf{p}_t(\mathbf{s}') \cdot \gamma_t((o, o'))) \tag{6}$$

$$\mathbf{p}_{t+1} = \mathbf{q}_t(L+1) \cdot \mathbf{p}^r_{t+1} + (1 - \mathbf{q}_t(L+1)) \cdot \mathbf{p}^s_{i+1} \tag{7}$$

Hence, the edge scores are responsible for shifting the probabilities of relevant objects from the previous time step. This is repeated over T steps. Finally, the probability distribution $\mathbf{p}_T$ and the reasoning relevance scores $\mathbf{q}_T(j)$ are used to aggregate the feature vectors from the nodes.

$$\mathbf{m} = \sum_{o \in V} \mathbf{p}_T(\mathbf{s}) (\sum_{j=0}^{L} \mathbf{q}_T(j) \cdot o^j) \tag{8}$$

Then this is concatenated with the question vector and fed to a feed-forward neural network for classification.

2.2 LCGN

The key component of LCGN is the construction of a contextualized representation for each object from the contextual information of its neighbors. This is obtained through iterative graph-attention (?) conditioned on the natural language commands $c = [\mathbf{c}_1, ..., \mathbf{c}_T]$. LCGN uses a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) encoder to represent the tokens of the question with the final hidden states of the bi-LSTM $\mathbf{H} = [\mathbf{h}_1; ..; \mathbf{h}_M] \in \mathbb{R}^{M \times l}$ and the final hidden states to represent the content of the entire question $\mathbf{q} = [\mathbf{h}_1; \mathbf{h}_M]$. At each time step, the question vector $\mathbf{q}$ transformed by a unique time-index matrix $\mathbf{W}_t$ is used to attend over the hidden states to produce $\mathbf{c}$ using the attention operation (Bahdanau et al., 2016).

$$\mathbf{c}_t = Attention(\mathbf{W}_t \mathbf{q}, \mathbf{H}) \tag{9}$$

$$Attention(\mathbf{y}, \mathbf{X}) = \mathbf{X}^T \alpha, \alpha = softmax(\mathbf{X}\mathbf{y})$$
$$\mathbf{X} \in \mathbb{R}^{N \times d}, \mathbf{y} \in \mathbb{R}^d \tag{10}$$

Hence, given the original object features $\mathbf{x}_i^{loc}$ which forms the nodes V in the graph , LCGN constructs contextual feature vectors $\mathbf{x}_{i_t}^{ctx}$ for each time step t using graph attention,

$$\mathbf{x}_{i_t}^{ctx} = \mathbf{W}[\mathbf{x}_{i_{t-1}}^{ctx}; (softmax(\mathbf{K}_{t-1}\mathbf{Q}_{t-1}^T)\mathbf{V}_{t-1}))_i] \tag{11}$$

where the queries $\mathbf{K}_{t-1}$ and $\mathbf{V}_{t-1}$ are acquired from the joint visual vector conditioned on the language command $\mathbf{c}_t$.

$$\mathbf{K}_{t-1} = [\mathbf{X}_{t-1}^{joint} \odot \mathbf{W}_{Kc}\mathbf{c}_t]\mathbf{W}_K \tag{12}$$

$$\mathbf{V}_{t-1} = [\mathbf{X}_{t-1}^{joint} \odot \mathbf{W}_{Vc}\mathbf{c}_t]\mathbf{W}_V \tag{13}$$

$$\mathbf{Q}_{t-1} = \mathbf{X}_{t-1}^{joint}\mathbf{W}_Q \tag{14}$$

where $\mathbf{X}_t^{joint} = [\mathbf{x}_{1_t}^{joint}; ...; \mathbf{x}_{N_t}^{joint}] \in \mathbb{R}^{N\times d}$. The joint visual vectors are acquired from the original and the contextual $\mathbf{x}_{i_t}^{joint} = [\mathbf{x}_i^{loc}; \mathbf{x}_{i_t}^{ctx}; \mathbf{x}_i^{loc} \odot \mathbf{x}_{i_t}^{ctx}]$. The original contextual features $\mathbf{x}_{i_0}^{ctx}$ are initialized with the local features.

Then the model pools the acquired contextual vectors and the original vectors through attention (Eqn. 10) with the question vector $\mathbf{q}$.

$$\mathbf{x} = Attention(\mathbf{Wq}, [\mathbf{X}^{loc}; \mathbf{X}_T^{ctx}]) \tag{15}$$

This pooled vector $\mathbf{x}$ is then passed through a final classifier of choice to produce an answer. In summary, LCGN produces a set of contextual vectors by allowing each object to attend to other objects with the help of the linguistic controller. The contextual vectors are iteratively refined for a fixed number of time steps. Then using the final contextual vectors, the question is used to find which objects are relevant in answering the question.

3 Dataset

We test our models on the GQA dataset (Hudson and Manning, 2019a) as both models of our focus used it as the primary dataset to report on their work. The GQA dataset focuses on real-world reasoning, scene understanding, and compositional question answering. It consists of 113K images and 22M questions of assorted types and varying compositionality degrees, measuring performance on an array of reasoning skills such as object and attribute recognition, transitive relation tracking, spatial reasoning, logical inference, and comparisons. In the dataset, the images are annotated with dense scene graphs (Johnson et al., 2015), representing the objects, attributes, and relations they contain.

Previous VQA datasets (and image captioning datasets) suffered from various problems. Impressive performances from early models were shown to be gained from taking advantage of the spurious correlations in the datasets, rather than actually learning to reason as humans do. In short, many previous benchmarks were biased, lacked the semantic compositionality, and did not provide tools to gain significant insights into models' performance and behavior (Antol et al., 2015; Das et al., 2019; Fisch et al., 2020). GQA dataset aims to overcome these issues by leveraging semantic representations of both modalities to mitigate language priors and conditional biases (Hudson and Manning, 2019a).

4 Concept Space Priored LCGN

Here we describe how the two models operate on the given concept vocabulary along with how the concept vocabulary is constructed. In equipping LCGN with the concept space prior, our primary focus was to retain the key features of the model as much as possible. This way, we can properly decorrelate the contribution from the prior from the contributions from the other assumptions of NSM. Hence, our objective was less on modifying LCGN to optimize its performance on a specific dataset using the prior, but more on measuring the improvements that can be achieved by changing how the prior is implemented in the model.

We identified 5 key features of LCGN:

1. (**F1**) contextual vectors are built from local vectors in an iterative manner

2. (**F2**) objects and commands are represented as d-dimensional vectors

3. (**F3**) relation information between the objects are implicitly controlled

4. **(F4)** the command vector controls the reasoning over graph only through the query and key vectors (Eqn. 12, 13)

5. (**F5**) the vector for classification is aggregated from the final contextual and local vectors through the attention from the question vector q (Eqn. 15)

These features were fundamental in constraining the design decisions of our concept space priored LCGN models: *Vector-Concept-Space (VCS)* and *Matrix-Concept-Space (MCS)* model. Both VCS and MCS use the same graph attention message passing algorithm of LCGN (F4) to build contextual vectors in an iterative manner (F1) as in Equation 11. Also, the aggregation of information from the graph for classification is done in the same manner (F5). Throughout this section, we will refer to the above 5 key features by noting the relevant constraint(s) (e.g. (F#)) in describing our models to further motivate our design decisions.

4.1 Concept Vocabulary

To create the concept vocabulary of size L, we initialize the known attributes from GQA with GloVe word embeddings (Pennington et al., 2014). Object names form their own concept, and we ignore the relations as LCGN does not inherently use them (F3). We create L clusters using k-means++ (Arthur and Vassilvitskii, 2007) clustering algorithm from the embeddings. Each concept (i.e. cluster) is represented with a prototype vector by averaging the embeddings of its members. We include the concept vocabulary in the appendix 7.

4.2 Vector Concept Space

Vector concept space (VCS) models represent each object and linguistic command as a d-dimensional vector (F2) through a linear combination of the prototype concept embeddings $\mathbf{r}_i$ (as described in 2.1.1) defined in the concept vocabulary. Since these models only use the prototype vectors and not the concept-member vectors $\mathbf{s}_j$'s (as described in 2.1.1), they are referred to as the "vector"-space models. This draws inspiration from the prototypical theory of concepts in cognitive science (Murphy, 2002). In the prototype theory, each category is represented with a prototype, and an object is classified as the category with the closest prototype to it. We present two VCS models: Local-VCS and Dynamic-VCS.

4.2.1 Local-VCS

The Local-VCS model maps the objects and the question to their relevant concepts only in the beginning. We simply map the local object features $\mathbf{x}_i^{loc}$'s transformed by a stem function f to the concept space through the attention-operation (Eqn. 10) on the prototype vectors $\mathbf{R} = [\mathbf{r}_1; \mathbf{r}_2; ..., \mathbf{r}_{L+1}] \in \mathbb{R}^{(L+1)\times d}$,

$$\hat{\mathbf{x}}_i^{loc} = \mathit{Attention}(f(\mathbf{x}_i^{loc}), \mathbf{R}) \tag{16}$$

We test different choices of stem f: linear, feed-forward neural network, and convolutional neural network. Also, we map each token embedding of the question to its closest concept through the attention-operation on the prototype vectors defined in C as done in NSM.

4.2.2 Dynamic-VCS

Dynamic-VCS is less constrained in mapping the visual and linguistic modalities to the concept space by being allowed to recompute the relevance scores of concepts for both modalities dynamically at each time step. Unlike Local-VCS, which had to correctly classify the object and question tokens all at the beginning, Dynamic-VCS can iteratively build contextual vectors and better align them to the concept space before classification.

The local object features $\mathbf{x}_i^{loc}$ are not mapped to the concept space, but only the contextual features $\mathbf{x}_i^{ctx}$ through the attention operation on the prototype vectors $\mathbf{r}_i$. Similar to Local-VCS, the vectors are transformed by a stem function f before being mapped to the concept space.

$$\hat{\mathbf{x}}_{it}^{ctx} = \mathit{Attention}(f(\mathbf{x}_{it}^{ctx}), \mathbf{R}) \tag{17}$$

This mapping operation is done at each time step for the newly constructed contextual vectors after the message passing algorithm (F4) in Equation 11.

The linguistic commands are produced by an LSTM-based encoder-decoder architecture (Sutskever et al., 2014). The token embeddings are not mapped to the concept space. Instead, the decoder is given the task of generating the commands in the concept space through unrolling. At each time step t, the hidden state of the decoder $\mathbf{u}_t$ is used to attend on the hidden states of the encoder $\mathbf{H} = [\mathbf{h}_1; \mathbf{h}_2; ...; \mathbf{h}_M]$.

$$\hat{\mathbf{u}}_t = \mathit{Attention}(\mathbf{u}_t, \mathbf{H}) \tag{18}$$

Then, the decoder hidden state and the collected attention vector are concatenated $\tilde{\mathbf{u}}_t = [\mathbf{u}_t; \hat{\mathbf{u}}_t]$. This concatenated vector $\tilde{\mathbf{u}}_t$ is used to attend on the prototype vectors $\mathbf{r}_i$ to produce the command $\mathbf{c}_t$ for T time steps. Hence, the linguistic controller maps each command to the most relevant concept at each time step.

4.3 Matrix Concept Space

The matrix concept space (MCS) model represents each object with the same d-dimensional vector as the VCS models but considers the relevance of an object to each concept independently of other concepts, comparing them with the concept members. Hence, we use the entire concept vocabulary ”matrix” for each concept by using the concept-member vectors $\mathbf{s}_j$ and not just the prototype vectors $\mathbf{r}_i$. This way of mapping to the concept space is akin to the exemplar theory of concepts in cognitive science (Murphy, 2002). In the exemplar theory, a category is represented as a set of typical exemplars, and an object is assigned membership to the category with the most similar exemplars.

The object features are mapped to the concept space only once in the beginning, similar to Local-VCS. First, the objects are transformed by a stem f.

$$\hat{\mathbf{x}}_i^{loc} = f(\mathbf{x}_i^{loc}) \tag{19}$$

We test the same choices of the stem as Local-VCS. Then the relevance score of an object for each concept is calculated by comparing the feature with the concept's prototype. The relevance score is converted to a probability through the logistic-sigmoid function.

$$\hat{p}_j(\hat{\mathbf{x}}^{loc}) = \sigma(\hat{\mathbf{x}}^{locT} r_j) \tag{20}$$

This probability $\hat{p}_j(\hat{\mathbf{x}}^{loc})$ is used to weigh the feature vector of the i-th object for the j-th concept $\mathbf{x}_{i_j}^{cpt}$. The feature vector $\mathbf{x}_{i_j}^{cpt}$ is calculated by attending on the concept members $\mathbf{s}_k$ with $\hat{\mathbf{x}}^{loc}$.

$$\mathbf{x}_{i_j}^{cpt} = \hat{p}_j(\hat{\mathbf{x}}_i^{loc}) Attention(\hat{\mathbf{x}}_i^{loc}, \mathbf{S}_j) \tag{21}$$

where $\mathbf{S}_j = [\mathbf{s}_1; ...; \mathbf{s}_{|C_j|}]$. Finally, all the feature vectors for each object are aggregated through the sum operation to produce the local feature vector $\mathbf{x}\tilde{^{loc}}_i$ to be used by the message passing algorithm (F1, F4).

$$\mathbf{x}\tilde{^{loc}}_i = \sum_j \mathbf{x}_{i_j}^{cpt} \tag{22}$$

The linguistic commands are prepared in the same way as Local-VCS by mapping each token embedding of the question to its closest concept by comparing with the prototypes $\mathbf{r}_i$.

5 Experiments

5.1 Goal of Experiment

It should be re-emphasized that the primary goal of our experiments are not to copy over the architectural decisions of NSM in implementing the concept space prior over to LCGN but to test the usefulness of the prior in the natural condition of LCGN. This way, we can identify any architectural decisions that are necessary to support the prior that were implicitly assumed in NSM. In other words, we are less interested in ”what” the prior is able to do, but more on measuring the effect of ”how” the prior is implemented can have on the performance of the model. Hence, the primary focus is on examining the discrepancies between the presented concept space priored models: Local-VCS, Dynamic-VCS, and MCS.

5.2 Experiment Setup

The GQA dataset provides ResNet (He et al., 2015) object features for each image extracted from Faster-RCNN (Ren et al., 2015) up to 100 regions along with the raw images and the scene graphs. We use these provided object features concatenated with their bounding box coordinates as the input to our models in all of our experiments. The GQA dataset provides many different dataset splits. In order to be consistent with LCGN, we train our models using the *train_balanced* set validating on the *val_balanced* set. We report on the accuracy of the models on the *testdev_balanced* with the best model according to the validation set. We used the same learning rate of 3e-4 and hyperparameter setting as the original LCGN model training for 25 epochs for the VCS models. All three models used the number of message passing iterations T=4. For MCS, we trained for 18 epochs with the same configuration for the other hyperparameters.

We introduce three additional hyperparameters in order to clearly understand the inner workings of the model. First hyperparameter is the choice of the stem function f for the local feature vectors as discussed in 4.2.1, 4.2.2, and 4.3. The function f can be a linear transformation, convolutional neural network (CNN), or a feed-forward neural network (FFN). CNN is a two-hidden-layered network with 2D convolution with kernel size of (3,3) and ELU (Clevert et al., 2016) non-linear activation. Output from CNN is flattened and linearly transformed. FFN is a two-hidden-layered net-

Model	Stem	GloVe Initialization	Fixed	val_balanced	testdev_balanced
LCGN (reported)	-	-	-	63.85	55.84
LCGN (ours)	-	-	-	63.40	55.18
Local-VCS	linear	T	T	62.00	**54.56**
	linear	F	F	57.7	49.81
	CNN	T	T	56.1	48.43
	FFN	T	T	**63.01**	53.98
Dynamic-VCS	linear	T	T	**63.01**	**55.11**
	linear	F	T	62.85	54.28
	linear	F	F	62.81	53.88
	linear	T	F	62.90	53.75
	**CNN	T	T	59.66	-
	FFN	T	T	62.95	55.01
	FFN	F	F	62.88	54.61
	FFN	T	F	62.98	54.99
MCS	linear	T	T	62.68	54.91
	**CNN	T	T	43.97	-
	FFN	T	T	**63.36**	**55.15**
	FFN	T	F	63.17	54.58
	FFN	F	T	62.77	53.92

Table 1: Accuracy (higher is better) is presented on *val_balanced* and *testdev_balanced* . "Stem" refers to the type of transformation used for the local object feature vectors before mapping to the concept space. "Fixed" refers to whether the model was allowed to change the concept embeddings. "GloVe Initialization" refers to whether the concept embeddings were initialized using the GloVe embeddings from the clustering result as described in 4.1. "T" refers to the condition being *true* and "F" refers to the condition *false*. **We could not evaluate these models on the $testdev$ set due to the saved files being lost. As seen in the table, the validation set is a good indicator of the performance on the test set. Thus, we decided to keep these models in the results.

work with residual connection from the original input and ELU non-linear activation. The second is the choice of initializing the concept embeddings (i.e. prototype vectors and concept member vectors) with the constructed concept vocabulary described in 4.1. If it is not initialized using the vocabulary, they are independently sampled from a standard normal distribution. This hyperparameter is used to measure how much of the model's performance can be improved through a well-structured concept space and to test whether the model is actually relying on the prior for reasoning. If the modes are able to do just as well with a randomly initialized concept space, this would point to the fact that these models are not relying on the concept space prior to produce the answers. The third is the choice of fixing the concept embeddings while training. If not fixed, the vectors will accumulate gradients from the supervision signals and optimize along with the other model parameters. This hyperparameter is used to examine whether it is possible to learn or better modify the concept space from the supervision signals.

5.3 Concept Vocabulary

We qualitatively found that k-means results in a few degenerative clusters that do not correspond to our intuition or are binary (i.e. cluster size of 1). These binary clusters are not necessarily too dissimilar from several other clusters that are internally somewhat loose (i.e. attributes in these clusters do not form a tight cluster). However, other than these clusters, many clusters were of good quality.

The number of clusters also had a big impact on the cluster quality. We experimented with several k number of clusters, with k ranging from 60 to 80. With different choices of k, we were able to improve the quality of some clusters, but it reduced the quality of others. As we did not see any improvement with these experiments, we use $k = 77$ as used in NSM.

5.4 Results

Table 1 summarizes the results of the three models. In this table, the model's performance on the *val_balanced* and the *testdev_balanced* are measured depending on different configurations of the hyperparameters discussed in 5.2: the different choices of the stem function f (column 2), whether the concept vocabulary was initialized using the constructed concept vocabulary from the GloVe embeddings (column 3), and whether the concept vocabulary was kept fixed during training (column 4). MCS performs the best with the accuracy of 63.36% on *val_balanced* and 55.15% on *testdev_balanced*, followed by Dynamic-VCS with the accuracy of 63.01% and 55.11%. Local-VCS performs much worse than the aforementioned two models with the accuracy of 63.01% and 54.56%. The VCS models perform much worse than MCS. This indicates that working with only the prototype vectors severely over-constrains the model. Each concept is composed of several members (i.e. exemplars) and the VCS models are not able to properly distinguish between them.

Another notable observation is that MCS matches what the original LCGN achieved (row 1), but it is not able to improve on the model. This result motivates two possible explanations. First, it indicates that it is not only important to operate on the concept space, but the mapping function that properly modulates the information from both modalities to the concept space is equally important. In NSM, each command vector is carefully mapped to the concept space by comparing with the prototype vectors to produce a probability distribution over which concept the command vector is talking about. Contrarily, in MCS (and thus in LCGN) such careful mapping is not possible as the command vectors are created by combining the concept space mapped token embeddings. Second, being able to think of the objects compositionally in terms of multiple concepts in parallel is crucial. MCS did so, but it had to represent each object as a single d-dimensional vector by aggregating the relevant concept embeddings. This sum operation might have led to some of the information needed for classification being lost. In NSM, only the relevance of each object is computed at each reasoning step. Only at the final aggregation step, the object features are flattened to a single d-dimensional vector for classification.

5.4.1 Effect of Stem Function

The choice of stem function had a significant effect on all three models. Particularly, the use of CNN as the stem function harmed the performance of all three This is most likely due to the limited receptive field not extending over the entire feature vector. This severely limited the kind of transformations the stem could perform. All three models were not able to fit properly, converging to a much lower training accuracy.

For MCS, FFN came out to be superior than the other choices. MCS requires a single feature

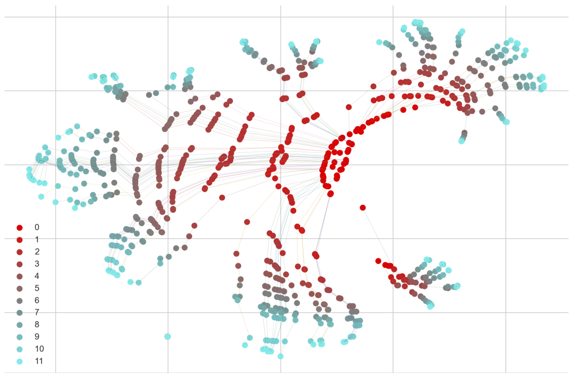

Figure 3: t-SNE (van der Maaten and Hinton, 2008) visualization of the Dynamic-VCS's concept space when it was not kept fixed during training for the first 11 epochs (red = epoch 1 and sky blue = epoch 11). The lines between the points indicate the trajectories of the individual concept embeddings over the training epochs. The concept space was initialized with constructed concept space using GloVe embeddings (Pennington et al., 2014).

vector that can be used to compare with multiple concepts in parallel. Hence, it requires a much more expressive stem than a linear transformation to construct such a vector. This indicates that being able to work with a concept space is contingent on the ability to construct invariant representations for the objects. If the features were invariant, the features of the objects from the same category would be very similar to each other. Hence, a linear transformation would have been enough to map these features to the same category.

For the VCS models, the linear stem function worked the best. This indicates that the VCS models' over-constrain makes them prone to overfitting. This again suggests that the additional structural prior of the MCS model that forces the model to think of object compositionally by calculating the object's concept relevance and the similarities to the individual exemplar vectors is important for operating in the concept space.

5.4.2 Effect of Concept Vocabulary

In all three models, equipping with the constructed concept space (i.e. rows with ”T” in column 3 of Table 1) showed an improvement over the randomly initialized concept space (i.e. rows with ”F” in column 3). This points to the fact that all three models are relying on the prior for reasoning by utilizing the information contained in the constructed concept space. If the models with the randomly constructed concept space were able to do as well, this would have pointed to the improper implementation of the concept space prior.

The discrepancy between the two conditions tells more about the nature of these models. Dynamic-VCS had the smallest gap of 0.19 between the two conditions (i.e. first two rows of Dynamic-VCS for *val*). This points to the fact that it was relying more on the other parts of the model in answering the questions rather than carefully utilizing the concept space. MCS showed a much larger gap of 0.59 between the two conditions, which indicates the contrary. MCS with randomly initialized concept space is able to do better than the best Local-VCS model equipped with a constructed concept space. This again confirms that the local-VCS model was over-constrained. In other words, the compositional structural prior instilled into MCS to use the prototype vectors to judge the relevance of an object for a concept independently of other concepts is an important ingredient of the concept space prior.

5.4.3 Effect of Learning Concept Space

Allowing the model to learn the concept space from the supervision signals had a negative effect on all three models (i.e. rows with ”F” vs ”T” in column 4 in Table 1). The clue as to why this is the case can be attained by looking at the learned concept embeddings. Figure 3 visualizes the trajectory of the concept space of Dynamic-VCS initialized using the concept vocabulary with t-SNE (van der Maaten and Hinton, 2008). We see that the semantic structure of the concept space drastically diverges from its initial structure in the early epochs (i.e. the points in the shade of red). Then the space starts to converge in the later epochs. We believe that this divergence phenomenon in the early training stage is due to the other parameters of the model being untrained. Hence, while the model attempted to learn to properly match both modalities to the concept space by changing the model parameters, the concept space had also moved from the supervision signals. Since the gradient only contains the first-order information and not the second-order interactions between the changes in the parameters of the model, this had led to the concept space diverging to a degenerate semantic structure. This shows that learning the concept space that generalizes well from supervision signals through gradients is a difficult task.

6 Related Work

In a way, the proposed models can be thought of as a type of memory networks (Weston et al., 2015; Sukhbaatar et al., 2015; Kumar et al., 2016). Memory networks have been widely used in natural language generation and dialogue learning where the ability to refer to the specific information given in the past is important (Weston, 2016; Ni et al., 2021). Memory networks operate over a discrete set of memories which can be long-term (used over more than one input) or short-term (created each time from the input). These memories are analogous to the concepts in our models. The input reads from the memories by selecting the most relevant ones. Usually, the reading operation is implemented using soft-attention to be trained using back-propagation. However, unlike our models, memory networks have an additional ”write” operation that explicitly updates the memories, rather than changing the embeddings through the gradient information.

7 Discussion and Conclusion

In this work, we study the importance of concept space prior in graphical LV models by modifying LCGN with the concept space prior. We propose three different concept space priored model architectures: Local-VCS, Dynamic-VCS, and MCS. The results point to the fact that concept space prior can be useful for the language and vision models, but only when supported by the right architectural mechanism. Specifically, the separation of the prototypes to represent the concept from the set of exemplars to decide how the concept is to be used for reasoning is necessary. While not explored in this work, we believe that the model would additionally need to know how the different concepts are related to each other to be able to operate on the concept space. This way, the existence of one concept would be able to constrain the other concepts that are mutually exclusive.

Also, reasoning on the objects compositionally with multiple concepts in parallel is crucial in being able to operate with the prior. In fact, the importance of compositional reasoning has been receiving large attention in the machine learning community in recent years (Lake et al., 2016; Loula et al., 2018) and the modern deep learning architectures are not compositional in nature (Baroni, 2019). The result from this work further motivates this movement for an intelligent system that is inherently compositional.

The ability to build invariant representations of objects is another ingredient needed for the prior. It is a known problem that a convolutional neural network is not view-point invariant (Sabour et al., 2017). Thus, the features used in the experiments had to be further transformed to be appropriately mapped to the concept space. In doing so, the VCS models had overfitted on the training set. As a future research direction, studies on the effect of using transformation or view-point invariant representations from the models such as Capsule-Network (Sabour et al., 2017) or Spatial Transformer Network (Jaderberg et al., 2016) are needed.

Learning the concept space that generalizes well from the supervision signals along with other parameters of the model was shown to be a difficult task. However, this result is not enough to conclude that such a concept space cannot be learned with the gradient information. There could be a better reading and writing operation that could provide a way to properly learn the concept space. In the future, a more extensive study on how the different reading and writing mechanisms affect the concept space learning is needed.

Analysis showed that many of these clusters were degenerate, and there are various ways to cluster the attributes. In the future, we wish to study the effect of fixing these degenerate clusters on the proposed models by aggregating these binary clusters with the cluster that is the closest to them. Additionally, qualitative cluster analysis showed that a few of the clusters were too large and therefore too general, while the rest of the clusters were rather small and tight. Although we have experimented with different centroid initializations, we have not yet found the right hyperparameters that could divide these large clusters into smaller and tighter ones. One future direction here could be neural clustering that might allow a better parameterization.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: visual question answering. *CoRR*, abs/1505.00468.

David Arthur and Sergei Vassilvitskii. 2007. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA. Society for Industrial and Applied Mathematics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate.

Marco Baroni. 2019. Linguistic generalization and compositionality in modern artificial neural networks. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 375(1791):20190307, Dec.

George Bealer. 1998. A theory of concepts and concepts possession. *Philosophical Issues*, 9:261–301.

Yoshua Bengio. 2019. The consciousness prior.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. 2020. Experience grounds language.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. Uniter: Universal image-text representation learning.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus).

Anubrata Das, Samreen Anjum, and Danna Gurari. 2019. Dataset bias: A case study for visual question answering. *Proceedings of the Association for Information Science and Technology*, 56:58–67, 01.

Adam Fisch, Kenton Lee, Ming-Wei Chang, Jonathan H. Clark, and Regina Barzilay. 2020. Capwap: Captioning with a purpose. *CoRR*, abs/2011.04264.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ronghang Hu, Anna Rohrbach, Trevor Darrell, and Kate Saenko. 2019. Language-conditioned graph networks for relational reasoning. *CoRR*, abs/1905.04405.

Drew A. Hudson and Christopher D. Manning. 2019a. GQA: a new dataset for compositional question answering over real-world images. *CoRR*, abs/1902.09506.

Drew A. Hudson and Christopher D. Manning. 2019b. Learning by abstraction: The neural state machine. In *NeurIPS*.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. 2016. Spatial transformer networks.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David Shamma, Michael Bernstein, and Li Fei-Fei. 2015. Image retrieval using scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Daniel Kahneman. 2011. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. 2016. Building machines that learn and think like people.

João Loula, Marco Baroni, and Brenden M. Lake. 2018. Rearranging the familiar: Testing compositional generalization in recurrent networks.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*.

Gregory L. Murphy. 2002. *The Big Book of Concepts*. MIT Press, Boston, Mass.

Jinjie Ni, Tom Young, Vlad Pandelea, Fuzhao Xue, Vinay Adiga, and Erik Cambria. 2021. Recent advances in deep learning based dialogue systems: A systematic survey.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 91–99, Cambridge, MA, USA. MIT Press.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, Melbourne, Australia, July. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

J. Weston, S. Chopra, and Antoine Bordes. 2015. Memory networks. *CoRR*, abs/1410.3916.

Jason Weston. 2016. Dialog-based language learning.

Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. 2018. Graph r-cnn for scene graph generation.

Appendix

Concept Vocabulary
0: "antique", "vintage"
1: "ornamental", "evergreen", "wicker", "blooming", "clay", "bamboo", "rustic", "deciduous", "potted", "hardwood"
2: "looking down", "up", "down", "looking up", "off"
3: "sleeveless", "lace", "ruffled", "quilted", "beaded"
4: "cloudy", "partly cloudy", "hazy", "overcast", "sunny", "foggy", "gloomy", "misty", "groomed", "cloudless"
5: "tight", "long sleeved", "wide", "short", "deep", "short sleeved", "long"
6: "tall", "thick", "wood", "square", "glass", "stone", "plain", "pine", "covered"
7: "lit", "fluorescent", "spraying", "neon", "illuminated", "unlit", "lighted"
8: "wavy", "wrinkled", "chubby", "skinny", "pointy", "braided", "curvy", "spiky", "shaggy", "puffy", "balding", "hairy", "crusty", "bushy", "bald", "curly", "crooked", "blond", "feathered", "wispy", "greasy", "brunette", "jagged", "furry", "immature", "speckled"
9: "iced", "pizza", "apple", "chocolate", "tasty", "delicious", "ripe", "juicy", "baked", "cheese", "banana", "sweet", "tomato", "strawberry", "fat"
10: "brass", "intricate", "exterior", "granite", "glazed", "polished", "chrome"
11: "breaking", "open", "tied", "shut", "wrist", "loose", "double decker", "running", "pulled back", "suspended", "chipped", "round", "closed", "broken", "rolled", "sealed", "edged", "crossed", "straight", "cut"
12: "skiing", "jumping", "skating", "skateboarding", "ski", "snowboarding", "surfing", "swimming"
13: "small", "huge", "heavy", "tiny", "massive", "vast", "giant", "large"
14: "public", "support", "office", "safety", "commercial"
15: "gravel", "asphalt", "railroad", "concrete", "paved", "unpaved", "elevated"
16: "wired", "connected", "barbed", "wire"
17: "narrow", "cobblestone", "spiral", "steep", "winding", "traffic"
18: "waving", "staring", "laughing", "smiling", "shirtless", "kneeling"
19: "blowing", "flat", "fallen", "driving", "upside down", "hitting", "flying", "crashing", "ocean", "palm", "alert", "hanging", "sliding", "overhead", "floating"
20: "trash", "garbage", "piled"
21: "patched", "bunched", "woven", "knit", "wrapped", "gloved", "knotted"
22: "inflated", "sturdy", "folded", "rimmed", "glossy", "upholstered", "inflatable", "oversized", "assorted", "cluttered", "tinted", "disposable", "fuzzy", "framed", "shiny", "colorful", "slanted", "floppy", "plush", "mesh", "styrofoam", "padded", "folding", "fancy", "neat"
23: "sleeping", "lying", "park", "riding", "walking", "parked", "busy", "barefoot", "sitting", "waiting", "resting"
24: "unpeeled"
25: "soda", "beer", "coffee", "wine"
26: "damaged", "faded", "mixed", "calm", "spread", "capital", "street", "old", "still", "roman", "ivory", "raised", "fire", "angry", "old fashioned", "abandoned", "turned", "rock", "powerful"
27: "funny", "curious", "ugly", "messy", "blurry", "unhealthy", "uncomfortable", "dirty", "unhappy", "sad", "dull", "comfortable"
28: "bronze", "silver", "gold"
29: "sheer", "bending", "muscular", "swinging", "outstretched", "melting", "artificial", "rippling", "curled", "spinning", "tilted", "bent", "dangling"
30: "athletic", "polo", "adidas", "nike"
31: "soap", "kitchen", "toilet", "bathroom", "cooking"
32: "murky", "stormy", "shallow", "choppy"
33: "indoors", "outdoor", "winter", "outdoors"
34: "tropical"

35: "sculpted", "stained", "carved", "carpeted", "octagonal", "painted", "marble", "brick", "decorated", "domed", "decorative", "recessed", "paneled", "ornate"
36: "frozen", "melted", "dried", "crumbled", "shredded"
37: "rocky", "sleepy", "leafy", "barren", "dusty", "grassy", "shaded", "manicured", "rugged", "lush", "dotted", "fenced", "forested", "wooded", "overgrown", "chalk"
38: "uncooked", "sprinkled", "boiled", "pepper", "steamed", "fried", "roasted", "eaten", "cooked", "grilled", "toasted", "seasoned"
39: "shaved", "halved", "trimmed"
40: "directional", "horizontal", "vertical", "tail"
41: "rubber", "aluminum", "metal", "steel", "stainless steel", "tin", "copper", "iron"
42: "vanilla", "coarse", "translucent", "creamy", "crisp", "fluffy", "foamy", "powdered", "crispy"
43: "male", "young", "elderly", "female"
44: "black and white", "brown", "blue", "red", "green", "pink", "yellow", "purple", "black", "white", "gray", "dark brown", "orange", "dark blue"
45: "shaped", "oblong", "elongated", "curved", "triangular", "cylindrical", "rounded", "rectangular", "arched", "angled", "sloped"
46: "cordless", "telephone", "wireless"
47: "sandy", "muddy", "dry", "warm", "rainy", "rough", "snow", "wet", "snowy"
48: "beige", "teal", "worn", "pastel", "patterned", "maroon", "jeans", "checkered", "plaid", "striped", "khaki", "flowered", "floral", "denim", "collared"
49: "reading", "standing", "reflected", "attached", "pointing", "sharp", "talking", "straw", "blank", "posing", "mounted", "drawn", "displayed"
50: "cloth", "wool", "silk", "leather", "cotton"
51: "light blue", "cream colored", "diamond", "smooth", "dark colored", "thin", "light brown", "dark", "crystal", "light colored", "bright", "rainbow colored", "soft", "tan", "pale", "shining", "light"
52: "raw", "grazing", "scarce", "abundant"
53: "incomplete", "patchy", "irregular", "sparse", "uneven", "dense"
54: "cardboard", "homemade", "plastic", "printed", "paper", "vinyl", "porcelain", "packaged", "handmade", "oriental", "fake", "stuffed", "miniature", "toy", "ceramic"
55: "birthday", "christmas", "wedding"
56: "computer", "analog", "portable", "electronic", "digital"
57: "sunlit", "unoccupied", "rippled", "polar", "grouped", "squatting", "scrambled", "perched", "crouching", "shadowed", "splashing", "crouched", "cushioned", "docked", "mowed"
58: "chinese", "foreign", "american", "french", "asian"
59: "chopped", "grated", "sliced", "peeled", "diced"
60: "browned"
61: "baby", "clean", "happy", "beautiful", "fine", "fresh", "little", "wild", "pretty", "real", "hard", "strong"
62: "half full", "clear", "full", "designed", "complete", "on", "having meeting", "new", "formal", "made", "written", "used"
63: "navy", "military"
64: "simple", "urban", "mature", "regular", "adult", "modern", "typical", "healthy"
65: "tennis", "professional", "baseball", "performing trick", "playing", "batting", "soccer"
66: "drinking", "water", "eating"
67: "high", "lower", "low", "upper"
68: "textured", "reflective", "opaque", "vibrant", "glowing", "protective", "abstract", "transparent"
69: "industrial", "electric", "gas", "power"
70: "license"

71: "lined", "vacant", "empty", "packed", "scattered", "filled", "burning", "stacked", "crowded"

72: "rotten"

73: "wii"

74: "tiled", "shingled"

75: "bare", "hollow", "tangled", "crumpled", "torn", "cracked", "weathered", "rusty", "twisted", "burnt"

76: "breakable", "calico", "ridged", "chain-link", "frosted", "corded", "wrinkly", "discolored", "tabby", "clumped", "unripe", "peeling", "caucasian", "leafless"