Curriculum Learning Effectively Improves Low Data VQA

Narjes Askarian
Dept. of Data Science and AI
Monash University

Ehsan Abbasnejad
Australian Institute for Machine Learning
The Univ. of Adelaide

Ingrid Zukerman and **Wray Buntine** and **Gholamreza Haffari**
Dept. of Data Science and AI
Monash University

Abstract

Visual question answering (VQA) models, in particular modular ones, are commonly trained on large-scale datasets to achieve state of the art performance. However, such datasets are sometimes not available. Further, it has been shown that training these models on small datasets significantly reduces their accuracy. In this paper, we propose a curriculum-based learning (CL) regime to increase the accuracy of VQA models trained on small datasets. Specifically, we offer three criteria to rank the samples in these datasets, and propose a training strategy for each criterion. Our results show that, for small datasets, our CL approach yields more accurate results than those obtained when training with no curriculum.

1 Introduction

Visual question answering (VQA) models are commonly trained on large-scale datasets to achieve the state of the art performance (Johnson et al., 2017a; Antol et al., 2015; Hudson and Manning, 2019). Modular VQA models, in particular, require large data sets for training. These models dynamically combine a number of neural networks according to a pre-specified layout (Andreas et al., 2016; Johnson et al., 2017b; Yu et al., 2018) to form a new larger network that produces an answer to an input question. The layout, or program, is generated for each question on the fly. As a consequence, the architecture of the resulting network varies according to the program.

Combining neural networks often leads to a wide and deep network. Training such a large-sized network with a varying architecture calls for a massive amount of labeled data, which is either expensive or very limited in many realistic settings. With insufficient data, a large and complex network can perform unsuccessfully. An example of this is our experience in training the VQA model by Johnson et al. (2017b) with only 20% of the CLEVR dataset (Johnson et al., 2017a). Our results showed only 54.24% accuracy compared to the accuracy of 96.90% on the full dataset according to the authors' report (Johnson et al., 2017b). Motivated by this experience, the work presented in this paper studies VQA in low data scenarios, and sheds light on the performance of current modular VQA models under data scarcity conditions. To the best of our knowledge, this is the first study to investigate VQA models in low-data regime.

Many approaches have been investigated to improve the performance of deep learning models when training on limited data, ranging from data augmentations (Zhang et al., 2019) and pre-training (Erhan et al., 2010) to semi-supervised learning (Kingma et al., 2014) and transfer learning (Raina et al., 2007). However, these works mostly deal with the scarcity of labeled data by assuming help from available unlabeled data, or by transferring knowledge from similar domains. Unlike them, our goal is to train a modular VQA model from scratch by using only a small amount of labeled data without using any other resources.

Specifically, we take the CL approach to tackle the problem of VQA models' low performance under low data conditions. Curriculum learning (Bengio et al., 2009) was introduced as a method to supervise the order in which data examples are exposed to the model. Our hope is to maximize the usage of training samples by performing supervision on the order of training data that are fed into the model.

The underlying idea of CL is to start learning from easy examples, and gradually consider harder ones, rather than using examples in a random sequence. To rank training examples from easy to hard, CL must define the concepts of easy and hard examples. Such a ranking is a key challenge in CL.

Many of the ranking criteria introduced in the CL literature are problem-specific heuristics (Liu et al., 2018) or automated measures based on model performance (Hacohen and Weinshall, 2019). In this paper, we propose and analyze the performance of three ranking criteria: (1) a length-based criterion, which considers longer questions as more complex than shorter questions, and ranks the examples in increasing order of their program length; (2) a criterion based on an answer hierarchy, which organizes all possible answers from coarse to fine; and (3) a criterion that relies on model loss for deciding about the hardness level of the examples and ranking them accordingly.

In addition to the ranking heuristics, in §5, we propose a CL training strategy for each criterion. We also argue that under CL training in low data regimes, a model is very susceptible to overfitting and poor generalization. Employing a regularizer is crucially important to prevent the model from becoming over-confident on the training data. We demonstrate that the proposed training strategies, when coupled with *L2-norm* regularization, lead to a significant improvement in performance, in some cases over 30% increase in accuracy.

We apply our approach to the model proposed by Johnson et al. (2017b) as a modular VQA model. The model originally consists of two main components: (1) a *program generator* that takes a question and generates a program; and (2) an *execution engine* that combines neural modules according to the program in order to create a network to produce an answer from the input image. Johnson et al. (2017b) demonstrate that the *program generator* can produce acceptable programs by training only on a small fraction of all possible programs ($\leq 4\%$). Thus, we focus on training the *execution engine* in a low-data setting and use ground-truth programs as input to the *execution engine*. To simulate a low data regime, we use four randomly chosen small subsets of the CLEVR dataset (Johnson et al., 2017a) for training. Our results show that our CL approach yields more accurate results than those obtained when training with no curriculum.

2 Background

Visual question answering is the task of inferring the answer by reasoning on the input question and image. Most of the current approaches map question-image pairs into a cross-modal common embedding space. A question is usually treated holistically in such approaches, thus the reasoning process is hard to explain (Tan and Bansal, 2019; Lu et al., 2019; Selvaraju et al., 2020).

In contrast, modular approaches perform visual reasoning by semantically parsing the question and generating a reasoning chain called a *program* (Andreas et al., 2016; Johnson et al., 2017b). The program shows the reasoning steps required for answering the question as a layout for the *modules*. The algorithm then combines the modules according to the program. Modules are small neural networks treated as single-task functions that are combined into a larger network to accomplish a complex job. The resulting network is *executed* on the input image to predict the answer.

Modular approaches naturally have a strong potential for interpretability. Hu et al. (2018) showed human evaluators can more clearly understand their modular VQA model compared to a non-modular model (Hudson and Manning, 2018). Thus, we are interested in studying modular models.

Similar to other VQA models, modular approaches call for a large amount of annotated data for both the semantic parser (program generator) and the executor. This issue has led to recent studies on sample efficient training strategies, ranging from multi-task learning (Hu et al., 2018) and active learning (Misra et al., 2018) to disentangling reasoning from vision and language understanding (Yi et al., 2018). For instance, Misra et al. (2018) propose an agent that, instead of operating on the training set, interactively learns by asking questions. Regarding the simulated low data setting in our work, efficient use of training data becomes extremely important. We employ curriculum learning in §4 and §5 as a method of making the best use of limited available data where a model can establish its understanding on simple concepts and gradually develop it by seeing harder examples over training.

3 VQA Model

In a VQA task, a model receives as input a pair $(\mathbf{x}, q)$ of image $\mathbf{x}$ and a question q about the image. The model learns to select an answer $a \in \mathcal{A}$ to the questions from a set $\mathcal{A}$ of possible answers.

The VQA model (Johnson et al., 2017b) includes two main components: a *program generator* $\mathcal{G}$ and an *execution engine* $\mathcal{E}$. The *program generator* predicts a program p to address a question q. The *execution engine* combines the modules according

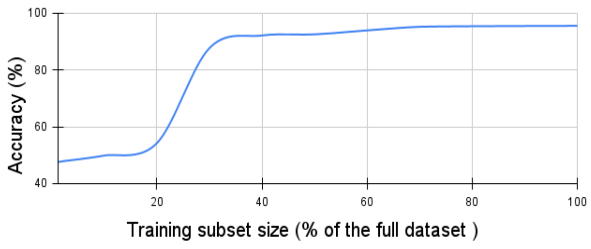


Figure 1: Accuracy of vanilla training of the *execution engine* on CLEVR `val` where trained on different-sized random subsets of the CLEVR `train` set.

to the program, and executes the obtained network on the image to produce an answer.

Johnson et al. (2017b) train the model using a semi-supervised learning approach. They demonstrate that the *program generator* can produce acceptable programs while training on only a small fraction of possible programs ($\leq 4\%$). To evaluate $\mathcal{E}$'s performance in a low data regime, we conducted a number of vanilla supervised training experiments with decreasing sized training sets. Note that we use ground truth program and image pairs as the input to $\mathcal{E}$ in all experiments. Figure 1 shows the best accuracy of each experiment on CLEVR's validation set while the *execution engine* is trained on a subset of the CLEVR's train set *e.g.,* 50% (See Figure 2 for some examples of the CLEVR dataset). The results verify *execution engine*'s poor performance on the small sized training subsets.

4 Curriculum Heuristics for VQA

Studies introduce various heuristics for measuring the hardness of examples. Some heuristics define hardness based on human judgment, in the sense that an example can be challenging for a machine if a human finds it difficult. Such criteria take features of examples into consideration such as *word frequency* and *sentence length* for texts (Spitkovsky et al., 2010; Platanios et al., 2019; Liu et al., 2018) and *shape complexity* for images (Bengio et al., 2009; Duan et al., 2020). The ordering of examples provided by these heuristics is task-dependent and does not change during training. In contrast, more general criteria determine the ordering of examples by incorporating the machine's response, *e.g.,* a teacher network supervises the learning process (Hacohen and Weinshall, 2019) or the progress of a model is taken into account (Kumar et al., 2010; Sachan and Xing, 2016; Zhou et al., 2021). In this study, we explore the heuristics described in the rest of this section.

4.1 Curriculum by program length

An intuitive measure of hardness for a VQA task is based on question length *i.e.,* longer questions are more complex to be understood and answered than shorter ones. This assumption has its root in the observation that a longer question generally involves understanding a larger number of objects and relations. We consider the length of the program corresponding to a question as an indicator of question length.

Under the program length curriculum, the network is fed with easy-to-hard ranked examples starting from shorter programs and gradually increasing programs' length.

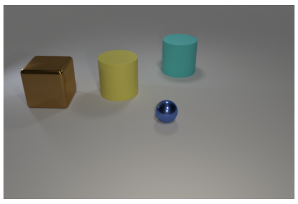
4.2 Curriculum by answer hierarchy

Investigating the learning process of $\mathcal{E}$ while training with IID data batching, we hypothesized the model implicit curriculum to be as follows: the model quickly learns to correctly predict the type of the answers, *e.g.,* color, size or digit. However, the more distinct values each type includes, the longer it takes for the model to distinguish them. For instance, the model needs a longer time to distinguish between eight different color values compared to *large* and *small* as the values of size. We also assume that the model struggles to identify visual features that are hard to detect, regardless of the number of distinct values, *e.g.,* whether the material of an object is *metal* or *rubber*.

Motivated by the above observations, we define another measure based on a hand-crafted answer hierarchy in order to shift the focus from questions to answers. The higher level in the hierarchy includes a coarser categorization of each answer type, and the answer types are vertically extended downward to finer classes of types. In other words, the direct link between an answer type and its values is interleaved with intermediate levels of abstraction, *e.g.,* digit at a lower level is divided into three groups, such as *'0'*, *'1'* and *many*. This classification splits into finer groups toward the bottom of the path. The details of the hierarchy are given in Appendix A of the supplementary material.

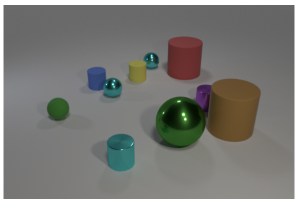
4.3 Curriculum by hard examples

The intuition of this heuristic is to focus training on the hard examples where the learner does not perform well and consequently the loss is high. The notion of hardness is considered dynamic, as a hard problem tends to be deemed easier while it is be-



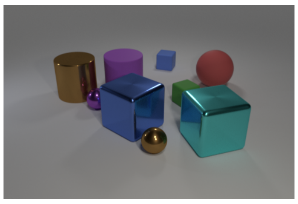
Easy Q: There is an object that is both right of the yellow rubber object and behind the large brown thing; what is its color? **A:** cyan

(A) Easy Question



Medium Q: What number of large objects are cyan metallic spheres or yellow spheres? **A:** 0

(B) Medium Question



Hard Q: What size is the metal block right of the brown metal thing right of the blue thing in front of the small blue rubber thing? **A:** large

(C) Hard Question

Figure 2: Examples of easy, medium and hard questions according to their H scores. The proposed heuristics do not always agree. According to the length-based heuristic, example A is harder than example B.

Hardness	**Epoch**					
	1	10	25	50	75	98
Easy	0.90	0.81	1.16	0.93	1.16	1.12
Medium	5.49	1.87	2.31	1.40	1.33	1.27
Hard	11.78	3.57	1.74	1.10	0.94	1.40

Table 1: Hardness scores at different epochs. The hardness scores decrease as training progresses.

ing understood. Following Zhou et al. (2020), we employ a dynamic hardness criterion based on the running average of *instantaneous hardness*, which is defined as the loss difference between two consecutive training iterations.

Let $(\mathbf{x}_i, p_i)$ be the ith image-program pair as a training example with the ground truth answer a_i. The instantaneous hardness $r_t(i)$ of $(\mathbf{x}_i, p_i)$ at time-step t is defined as follows:

$$r_t(i) = |\ell_t(a_i - \mathcal{E}(\mathbf{x}_i, p_i; w_t)) - \ell_{t-1}(a_i - \mathcal{E}(\mathbf{x}_i, p_i; w_{t-1}))| \quad (1)$$

where t represents training epochs.

The hardness score of an example is obtained by recursively computing a running average over instantaneous hardness, which reflects the dynamics of hardness,

$$H_{t+1}(i) = \begin{cases} \gamma \times r_t(i) + (1-\gamma) \times H_t(i) & \text{if } i \in S_t \\ H_t(i) & \text{else} \end{cases} \quad (2)$$

where $\gamma \in [0, 1]$ is a discount factor, and $S_t \subseteq \{(\mathbf{x}_1, p_1), ..., (\mathbf{x}_N, p_N)\}$ is a subset of the training set selected at each training step according to a sampling strategy. We employ the strategy of Johnson et al. (2017b), which uses a probability function based on the hardness score H. This function favors harder examples so long as the probability of selecting easy examples is not zero.

Once a sample is used to train the model, its H score becomes small and it stays low relative to the other samples. Thus samples' H score converges during training and remains consistent. This gives the unselected samples a higher chance to be selected by the sampling function in the future steps. Figure 2 shows three samples with low, medium and high H scores (denoted as easy, medium and hard questions) at the first iteration and Table 1 lists their corresponding H scores during training. It is clear that the H score is decreasing over training until convergence.

5 Curriculum Learning for VQA

We describe now our training procedure. A generic curriculum learning requires a model M and a training dataset D as inputs. It also requires the existence of a hardness criterion N, a curriculum scheduler E, a selection function L, and a performance measure P.

According to traditional curriculum learning, at every training iteration, the scheduler E decides when to update the curriculum. Curriculum learning is applied on top of the conventional training loop in machine learning. The output of each training loop is usually the model's performance measure, which may be used by the scheduling function L to specify the appropriate moment for modifying the curriculum. The scheduler can also decide merely based on the number of training iterations. A curriculum update typically includes re-ranking training examples according to the hardness criterion N. In the next step, the algorithm selects

Algorithm 1 Scheduled Training with Curriculum

```
1: $\mathcal{E}$: *execution engine*
2: $\{(\mathbf{x}_i, p_i, a_i)\}_{i=1}^{n}$: training examples
3: $\gamma$: $\in [0, 1]$, discount factor for reducing subset size
4: $T$: number of iterations
5: $T_0$: number of warm-starting iterations
6: procedure HEMTRAINING
7:     for $t \in \{1, ..., T\}$ do
8:         if $t \leq T_0$ then                         ▷ Phase1: Warm-starting
9:             $S_t = [n]$
10:        else                                         ▷ Phase2: Hard example mining
11:            for $i \in \{1, ..., n\}$ do
12:                $p_i = H_t(i) + C_t(i)$
13:            end for
14:            Normalise($p_i$)
15:            $S_t \leftarrow$sample $k_t$ district elements from $Categorical(\vec{p})$
16:            $w_t \leftarrow w_{t-1} + \pi\left(\nabla_w \sum_{i \in S_t} \ell(a_i, \mathcal{E}(p_i, x_i; w_{t-1}))\right)$
17:        end if
18:        Compute $r_t(i)$ for $i \in S_t$ using Eq. (1)
19:        Update $H_{t+1}(i)$ using Eq.(2)
20:        $k_{t+1} \leftarrow \gamma_k \times k_t$
21:    end for
22: end procedure
```

a subset D^* of the training set D, which will be used by the model in the next round of training. The selection function SF can utilize different approaches, *e.g.,* weighting (Liang et al., 2016; Zhou et al., 2020), sampling (Zhou et al., 2021) or batching (Yong Jae Lee and Grauman, 2011).

Training by length-based curriculum. We design a CL training strategy for the length-based curriculum by equipping the CL training with a batching method as the selection function and a linear paced scheduler. The scheduler controls the curriculum update at a linear pace, *i.e.,* a hyper-parameter specifies the number of iterations for learning a curriculum.

Training by answer hierarchy curriculum. Our proposed training algorithm for the answer hierarchy curriculum takes advantage of a simple self-paced scheduler based on the model performance. Specifically, the scheduler updates the curriculum where the normalized difference of accuracy between two consecutive iterations goes higher than a predefined threshold.

Training by hard examples curriculum. This training strategy suggests training the model in two phases. The first phase is a warm-up phase, where the model sweeps all training examples. The next phase is curriculum training, where the model ranks the examples according to their hardness and learns a selected subset of them.

Algorithm 1 summarizes our training approach. To encourage diversity, we add a submodular optimization C to the hardness score in line 12, which is inspired by Zhou and Bilmes (2018). Since this can be any submodular function, we choose a function based on the similarity between examples,

$$\max_{S_t} \sum_{i \in S_t} H_t(i) + \lambda_t C(S_t) \tag{3}$$

where $C(S_t) = \sum_{i,j \in S_t} w_{i,j}$ and $w_{i,j}$ represents the similarity between example i and j. The preference for diversity can be controlled by λ_t. We gradually reduce it during training to further focus learning on hard examples. The input to C is a representation of a data point that can be a fusion of both text and image modalities. For this, we use the output of the model's penultimate layer as the representations of the examples.

Instead of deterministically choosing the top k samples based on H, we randomly select the examples for the next round of training with the probability $p_{t,i} \propto f(H_{t-1}(i))$ where $f(.)$ is a non-decreasing function, similar to Zhou et al. (2020). This probability function favors hard examples, yet selecting easy ones is possible. At early training, when the H scores are poorly estimated, $f(.)$ should encourage exploration, and move toward more exploitation as training progresses and H estimation is becoming more accurate. We balanced the trade off between exploration and exploitation using the upper confidence bandit (UCB) algorithm, similar to Auer et al. (2003) and Zhou et al. (2020),

$$f(i,t) = Normalized\Big[H_t(i) + c\sqrt{\log T / N_t(i)}\Big]$$

where T is the number of iterations, and $N_t(i)$ is the number of times that the ith sample has been selected prior to time step t. UCB controls the degree of exploration by the hyper-parameter c which we set as 0.001 in our implementation.

5.1 Improved Curriculum Learning

The idea of learning the answers in a non-random ordering as what happens in CL has been shown to be helpful for the learning process in many cases. However, this idea has one essential deficiency. It focuses on a particular subset of questions early and is not exposed to a diverse set of questions. When a new question arrives, the algorithm struggles to adjust to it, as the learned representations fit the previous questions. This problem exacerbates in low data settings. Many studies highlight the importance of selecting a diverse set of examples as a solution to this issue (Sachan and Xing, 2016; Zhou and Bilmes, 2018), and the CL algorithm generally benefits from diversity in training examples. However, as confirmed by our experiments (§6.4), it does not prevent the model from overfitting. We, therefore, explore the effect of other techniques of regularizing such as dropout and L2-norm.

6 Experiment

We use our implementation of the *execution engine* model (Johnson et al., 2017b). A vanilla training of the model posts the lowest threshold of the performance in our setting. We also implemented and compared the three heuristics for the hardness criterion: *program* length (§4.1), *answer hierarchy* (§4.2) and *hard example* (§4.3). The *length-based* curriculum can be seen as a baseline to the *answer hierarchy* criterion, while both of them play the role of baseline for the *hard example* curriculum. We do not compare with the state of the art, because the goal of our paper is to study VQA in a low-data regime, and to the best of our knowledge, there is no other work that conducts similar research. Thus, we focus on improving the performance of our baseline models.

We assessed our baselines under the following conditions: i) **No-Reg** when no regularizer is applied. ii) **Dropout** when we apply dropout technique to the final linear layer (classification layer) in $\mathcal{E}$. iii) **L2-norm** when *L2-norm* regularizer is applied as a weight decay to the optimizer.

6.1 Dataset

We evaluate our approach on the CLEVR dataset (Johnson et al., 2017a), which provides a training set with $70k$ images, $\sim 700k$ $(\mathbf{x}, q, a)$ tuples and 32 answer classes. To simulate a low-data regime, we randomly sample four subsets of different sizes from CLEVR `train`. The size of the subsets are 5%, 10%, 15% and 20% of the full `train`set, which contain $35k$, $70k$, $105k$, and $140k$ $(\mathbf{x}, q, a)$ tuples respectively. We call these subsets s-CLEVR$_p$, where p denotes the percentage of the subset size wrt `train`, *e.g.,* s-CLEVR$_{15}$ refers to the subset of size 15% of `train`. As CLEVR `train` and CLEVR `val`(the evaluation set) have similar answer distributions, to perform a fair comparison, it is important that the sampled subsets also have similar answer distributions. Our evaluation is conducted on the `val`split, which contains $\sim 150k$ questions and $15k$ unique images.

6.2 Baselines

No-CL is used as the vanilla baseline where the *execution engine* is trained with an IID sampling on s-CLEVR subsets without any curriculum. In other words, the model sees all examples in the training set at every iteration.
Length-CL follows a linear paced scheduler when training the *execution engine*under the length-based curriculum (4.1).
AnswerH-CL makes use of a self-paced scheduler based performance measurement and the answer hierarchy curriculum (4.2). The curriculum updates if the changes in normalized accuracy between two consecutive iterations are higher than a pre-specified threshold. A batching function selects the sampled for every training iteration.
HardEx-CL uses the hard example heuristic 4.3 as the criterion of ranking data and follows the algorithm 1 for training. Unless stated otherwise, we use HEM-CL in all ablation analysis experiments.

6.3 Implementation Details

The *execution engine* uses the images features from $conv4$ of ResNet-101 (He et al., 2016) pretrained on ImageNet (Deng et al., 2009). We use Adam (Kingma, 2015) with a fixed learning rate of $1e\text{-}4$ to optimize the first three baselines and a cyclic cosine annealing learning-rate schedule to optimize HEM-CL. In the case of the experiments that use *L2-norm*, a weight decay of $5e - 4$ is added to the ADAM optimizer. We also use dropout $= 0.5$ for some experiments.

6.4 Results and Discussion

Curriculum heuristics' effect. We evaluate the impact of our proposed training strategies with the three heuristics by looking at their performance on CLEVR `val`in Table 2 while training on s-CLEVR subsets. As the table shows, using the **length-based curriculum** yields poor accuracy almost in all cases of s-CLEVR training subsets with and without regularization. An explanation for this could be overfitting. As mentioned, overfitting is a serious challenge in low data training.

According to our analysis, there is a high chance for the model to overfit some modules because they are more likely to appear in the first positions of a program. Figure 3 depicts the frequency of modules' appearance in various positions of programs in about $28k$ programs. These modules are commonly related to an *anchor object* in a question,

Method	**No-Reg**				**Drop-out**				**L2-norm**			
	5%	10%	15%	20%	5%	10%	15%	20%	5%	10%	15%	20%
No-CL	46.91	48.77	49.68	51.25	46.94	48.36	49.67	49.92	46.71	50.25	52.20	54.34
Length-CL	46.55	46.67	47.83	48.12	46.68	47.33	47.61	47.71	47.89	49.65	50.98	51.50
AnswerH-CL	47.42	48.59	49.73	51.65	47.43	47.73	48.60	50.24	48.62	49.03	48.70	48.95
HardEx-CL	47.93	50.04	51.97	53.14	48.80	49.94	51.69	56.29	48.95	51.49	53.27	**87.62**$\pm$1.3

Table 2: The *execution engine* accuracy (%) on CLEVR valwhen training on s-CLEVR$_5$, s-CLEVR$_{10}$, s-CLEVR$_{15}$ and s-CLEVR$_{20}$ with three different choices of curriculum. The length-based (**Length-CL**) and answer hierarchy (**AnswerH**) curriculum does not improve the performance while hard example (**HardEx-CL**) outperforms the vanilla baseline (**No-CL**) in all experiments.

where other objects are described by their relation to this object, *e.g., the yellow thing* is the anchor in the question *"What is the size of cube to the right of the yellow thing"*. To identify *the cube* and determine its *size*, one must find *the yellow thing*, and attend to the objects on its left side. Since objects are normally described by attributes such as color, size and material, attribute-related modules tend to appear at the beginning of a program.

Ranking programs by their length makes the model focus on a limited number of modules during early training, which increases the chance of overfitting. The model thus struggles with learning other modules when they appear later in longer programs. According to the results, dropout and L2 regularizations do not effectively prevent overfitting where the curriculum forces the model to over-concentrate on such structural biases in data.

Answer hierarchy curriculum makes a marginal improvement on some subsets particularly s-CLEVR$_5$. **Hard example curriculum** produces impressive results, improving the baselines in all cases. The result verifies the effectiveness of emphasizing hard examples in low data regimes where due to the limited size of data and its large capacity, a deep network tends to memorize easy data points without actually learning a pattern. Forcing the model to focus on hard examples induces a form of implicit regularization. Additionally, the self-pacing feature of the curriculum allows the algorithm to update the curriculum based on its progress.

Table 2 also shows that **HardEx-CL** method does not produce the best accuracy per se. Regarding that the table reports the average results, it is noteworthy to mention that the best accuracy we achieved in the case of **HardEx-CL** is 88.83 score in accuracy where the weights are uniformly initialized and L2-norm is used for regularization. In

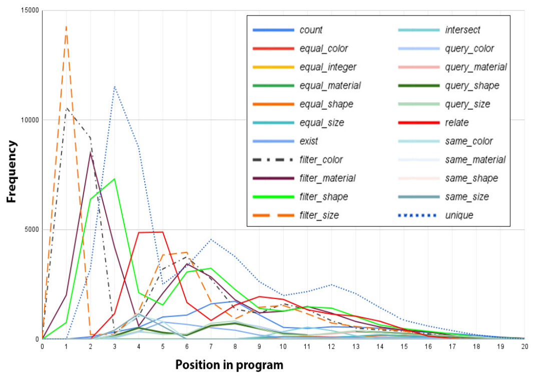


Figure 3: Frequency of modules appearance in different positions of programs. Some modules are more likely to appear at the first positions.

fact, the regularization causes a huge rise in accuracy. The next paragraphs look into the reasons that our regularization choice effectively boosts the **HardEx-CL** approach.

Regularization impact. To investigate the impact of different regularizers we conducted ablation studies by applying L1-norm in addition to L2 and drop-out regularization. Table 3 shows that in contrast to dropout and L1-norm, using L2 regularization results in improved performance in almost all experiments. To investigate the role of L2 regularization in CL training, we conducted an ablation experiment on the selected examples in **HardEx-CL** algorithm with and without L2-norm. First, we record the hardness measures of selected examples at every epoch $H_t(i)$ and split the range of measures into three categories, *easy*, *medium* and *hard*. The population distribution of examples by their hardness measure has a long tail. This long tail is excluded from the splitting and categorized as *very hard*. We then calculate the proportion of each category in the selected examples at 100 epochs as plotted in Figure 4.

These plots provide insight into the behavior of L2 regularization. Specifically, we observe that ex-

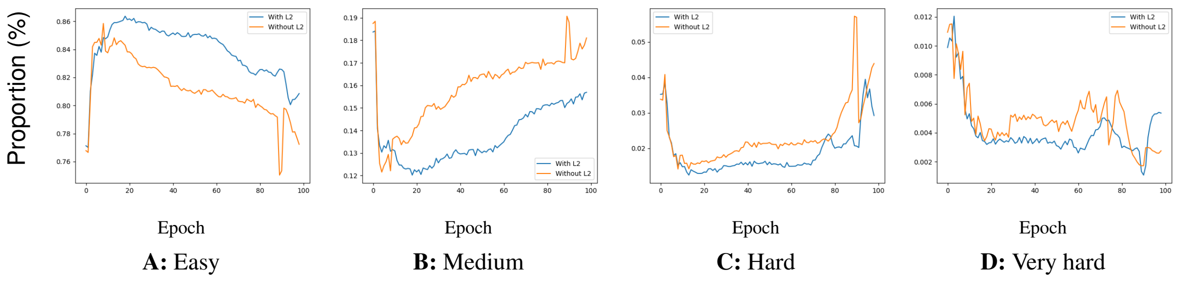


Figure 4: The proportion of different hardness categories in selected examples at 100 epoch in case of with and without L2 regularization. The regularization prevents forgetting by forcing the algorithm to incorporate more easy samples in the training set.

	No-Reg	Drop-out	L1-norm	L2-norm
No-CL	51.25	49.92	45.12	54.34
CL	53.14	56.29	46.79	86.65

Table 3: The impact of different regularizer on HardEX-CL accuracy when training on s-CLEVR$_{20}$.

cept for the easy category, the proportion of examples from other categories is higher for all epochs. It can be explained by the fact that **HardEx-CL** algorithm draws model attention to hard examples during training. As the model is learning the examples, their corresponding hardness measure is decreasing so that they finally are learned and considered as easy. Without using L2 regularization the model overly focuses on learning hard examples and as a consequence forgets the learned patterns of easy examples. L2-norm protects the model from forgetting such patterns by incorporating in loss and forcing the sampling function to also samples more from easy category.

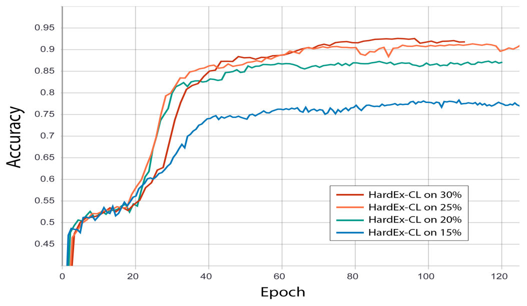


Figure 5: The accuracy of **HardEx-CL** algorithm on CLEVR `val`where *execution engine* weights is uniformly initialized and trained on s-CLEVR$_{15,20,25,30}$.

Why is there a jump in the accuracy of HardEx-CL with L2 regularization when training on s-CLEVR$_{20}$? Looking closely at the learning curve of *vanilla* training in Figure 1 reveals that the *execution engine* performance experiences a jump using training subsets larger than 20%. Different shapes of learning curves are defined in learning theory (Ebbinghaus, 1913; Bills, 1934). The S-curve that we can see here is the idealized general form of learning where the learner slowly accumulates small steps at first followed by a steep up stage with larger steps and the smaller steps successively occur to level off the curve. Due to lack of data, we do not see this performance gap when training on s-CLEVR$_{5-20}$. L2 regularization, however, stimulates the jump to happen earlier in **HardEx-CL**. To investigate it further, we run **HardEx-CL** with four training subsets of different sizes including 15%, 20%, 25% and 30% and report the accuracy on CLEVR `val`in Figure 5. All settings are similar to **HardEx-CL** with L2-norm in Table 2 except the weights are uniformly initialized. From these experiments, we observe the jump in the training set for even s-CLEVR$_{15}$ other than larger subsets. This shows the tipping point in the training can accrue earlier depending on the algorithm and settings.

7 Conclusion

This paper studied VQA in low data settings and shed light on the low performance of VQA models under the data scarcity condition. To improve the performance, we propose three curriculum learning approaches based on length, answer hierarchy, and hard examples. We also stressed the problem of overfitting and poor generalization that becomes crucially important in the absence of sufficient data. We explored the effect of using generalization techniques on a models' performance in low data regimes. Our results show that the proposed CL algorithms outperform the baseline in many cases while fail in some others. However, the algorithms when coupled with L2 regularization lead to improvements.

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *IEEE international conference on computer vision*, pages 2425–2433.

Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. 2003. The Nonstochastic Multiarmed Bandit Problem. *SIAM Journal on Computing*, 32(1):48–77.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *International Conference on Machine Learning*, ICML '09, pages 41–48, Montreal, Quebec, Canada. Association for Computing Machinery.

Arthur Bills. 1934. *General experimental psychology*. Longmans Psychology. Longmans, Green and Co.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. ISSN: 1063-6919.

Yueqi Duan, Haidong Zhu, He Wang, Li Yi, Ram Nevatia, and Leonidas J. Guibas. 2020. Curriculum DeepSDF. In *Computer Vision – ECCV*, Lecture Notes in Computer Science, pages 51–67, Cham. Springer International Publishing.

Hermann Ebbinghaus. 1913. *Memory: A Contribution to Experimental Psychology*. Annals of Neurosciences, Teachers College, Columbia University.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. 2010. Why Does Unsupervised Pre-training Help Deep Learning? In *International Conference on Artificial Intelligence and Statistics*, pages 201–208. JMLR Workshop and Conference Proceedings.

Guy Hacohen and Daphna Weinshall. 2019. On The Power of Curriculum Learning in Training Deep Networks. In *International Conference on Machine Learning*, pages 2535–2544. PMLR.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. ISSN: 1063-6919.

Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2018. Explainable neural computation via stack neural module networks. In *European conference on computer vision (ECCV)*, pages 53–69.

Drew A. Hudson and Christopher D. Manning. 2018. Compositional Attention Networks for Machine Reasoning. In *International Conference on Learning Representations*.

Drew A Hudson and Christopher D Manning. 2019. GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6700–6709.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017a. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2901–2910.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017b. Inferring and executing programs for visual reasoning. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2989–2998.

Diederik P. Kingma. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA. Conference Track Proceedings.

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3581–3589, Montreal, Canada. MIT Press.

M. Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-Paced Learning for Latent Variable Models. In *Advances in Neural Information Processing Systems (NIPS)*, volume 23.

Junwei Liang, Lu Jiang, Deyu Meng, and Alexander Hauptmann. 2016. Learning to detect concepts from webly-labeled video data. In *International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 1746–1752, New York, New York, USA. AAAI Press.

Cao Liu, Shizhu He, Kang Liu, and Jun Zhao. 2018. Curriculum learning for natural answer generation. In *International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 4223–4229, Stockholm, Sweden. AAAI Press.

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

Ishan Misra, Ross Girshick, Rob Fergus, Martial Hebert, Abhinav Gupta, and Laurens van der Maaten. 2018. Learning by Asking Questions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11–20. ISSN: 2575-7075.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. 2019. Competence-based Curriculum Learning for Neural Machine Translation. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1162–1172, Minneapolis, Minnesota. Association for Computational Linguistics.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. 2007. Self-taught learning: transfer learning from unlabeled data. In *International conference on Machine learning (ICML)*, ICML '07, pages 759–766, New York, NY, USA. Association for Computing Machinery.

Mrinmaya Sachan and Eric Xing. 2016. Easy Questions First? A Case Study on Curriculum Learning for Question Answering. In *54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 453–463, Berlin, Germany. Association for Computational Linguistics.

Ramprasaath R. Selvaraju, Purva Tendulkar, Devi Parikh, Eric Horvitz, Marco Tulio Ribeiro, Besmira Nushi, and Ece Kamar. 2020. SQuINTing at VQA Models: Introspecting VQA Models With Sub-Questions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10003–10011.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From Baby Steps to Leapfrog: How "Less is More" in Unsupervised Dependency Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, California. Association for Computational Linguistics.

Hao Tan and Mohit Bansal. 2019. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, Hong Kong, China. Association for Computational Linguistics.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. 2018. Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. *Advances in Neural Information Processing Systems*, 31.

Yong Jae Lee and K. Grauman. 2011. Learning the easy things first: Self-paced visual category discovery. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1721–1728, USA. IEEE Computer Society.

Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. 2018. Mattnet: Modular attention network for referring expression comprehension. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1307–1315.

Xiaofeng Zhang, Zhangyang Wang, Dong Liu, and Qing Ling. 2019. DADA: Deep Adversarial Data Augmentation for Extremely Low Data Regime Classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2807–2811. ISSN: 2379-190X.

Tianyi Zhou and Jeff Bilmes. 2018. Minimax Curriculum Learning: Machine Teaching with Desirable Difficulties and Scheduled Diversity. In *International Conference on Learning Representations, (ICLR)*, Vancouver, BC, Canada.

Tianyi Zhou, Shengjie Wang, and Jeff Bilmes. 2021. Curriculum Learning by Optimizing Learning Dynamics. In *International Conference on Artificial Intelligence and Statistics*, pages 433–441. PMLR.

Tianyi Zhou, Shengjie Wang, and Jeffrey Bilmes. 2020. Curriculum Learning by Dynamic Instance Hardness. In *Advances in Neural Information Processing Systems*, volume 33, pages 8602–8613. Curran Associates, Inc.

Supplementary Material

We describe more key implementation details of our work in the ensuing sections.

Appendix A: Curriculum by answer hierarchy

As mentioned in §4.2, the answer hierarchy, shown in Figure 6, classifies the answers at different hierarchical levels. Specifically, we defined intermediate levels between answer types and their values. The intermediate levels are employed as the higher level pseudo answers to the questions. According to the curriculum, the algorithm maps the true answer to the higher levels pseudo answers in order to gradually guide the predicted answers from a coarse level to a more specific one. When the scheduler decides to update the curriculum, several nodes are expanded to the next level, *i.e.,* the model is exposed to the finer level of an answer type. We do not force the curriculum to simultaneously expand all of the nodes that are at a similar level of the hierarchy. Instead, we assign a number to every node that determines the expansion time in terms of curriculum update round. Specifically, a node is expanded when the count of the curriculum update is matched with its assigned number. For instance, the node **'size'** is expanded to its children **'small'** and **‘large’** in the second round of curriculum update if number **2** is assigned to the node **‘size’**. This provides a degree of freedom for the algorithm to gradually learn the answers. Although we statically specify these numbers in our algorithm, they can be implemented as learnable parameters, which we leave to future work. Learning expansion times helps the model move the curriculum further at its pace.

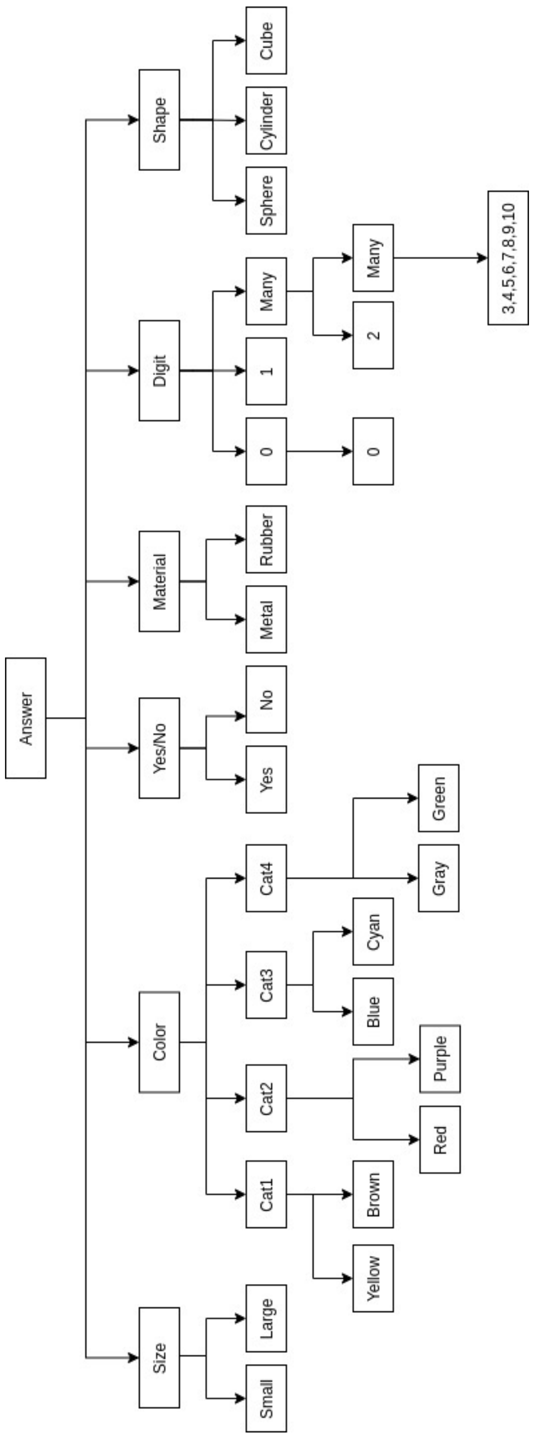


Figure 6: A schematic view of the answer hierarchy used as the base of a curriculum.