Testing Egunean Behin Visual Question Answering Dataset with BLIP

Julen Etxaniz
University of the Basque Country (UPV/EHU)
jetxaniz007@ikasle.ehu.eus

Abstract

Egunean Behin is a new visual question answering dataset based on Egunean Behin game. It consists of three types of questions from the game: figures, cubes and maze. Questions require counting figures, colors, cubes and understanding the dimensions of the pictures. Compared to other datasets that mostly use real-world images, all the artificial images and questions are generated automatically. There are multiple questions for each image and each question has one correct and two wrong answers. It can be used to evaluate VQA models in an out of domain zero-shot setting. In this work we test state-of-the-art BLIP model and show that these types of questions are very challenging. Accuracy drops a lot when compared to finetuning accuracy on VQA dataset.

1 Introduction

Egunean Behin is a popular Basque quiz game. The game consists on answering 10 daily multiple choice questions. Questions were translated to English because state-of-the-art VQA models are mainly trained on English questions. All the images and questions were generated automatically in two steps. First, we generate as many images as we want. Then, we generate multiple questions for each image.

Three types of questions from the game were selected: figures, cubes and maze. Questions require counting figures, colors, cubes and understanding the dimensions of the pictures. Each question has one correct and two wrong answers. These can be used for multiple choice question answering.

This dataset can be used to test VQA models in an out of domain setting. It could also be used to finetune a model to answer these types of questions if enough data is generated. The available code can be used to generate as many images as necessary.

In this work we will test state-of-the-art BLIP (Li et al., 2022) model on this dataset in a zero-shot setting. This will show the difficulty of each question type and the ability of the model to generalize to different types of images.

We will first describe related work on vision-language pre-training and bootstrapping. Then we will introduce the model architecture of BLIP, together with dataset bootstrapping, finetuning and testing methods. Next, we will describe images and questions in the Egunean Behin dataset. Finally, we will compare the results and extract some conclusions.

2 Related Work

This section explains related work on vision language pre-training and dataset bootstrapping. These are important to understand the limitations of previous methods and the solutions proposed in BLIP (Li et al., 2022).

2.1 Vision-language Pre-training

Vision and language are two of the most fundamental methods for humans to perceive the world. An important goal of AI has been to build intelligent agents that can understand the world through vision and language inputs, and communicate with humans through language.

Vision-language pre-training has emerged as an effective approach to achieve this goal. Deep neural network models are pre-trained on large scale image-text datasets to improve performance on downstream vision-language tasks, such as image-text retrieval, image captioning, and visual question answering.

Models are commonly pre-trained before they are fine-tuned on each task. Fine-tuning involves additional training of the pre-trained model, using data from the downstream task. Without pre-training, the model needs to be trained from scratch on each downstream task, which leads to worse performance.

Despite the success of vision-language pre-training, existing methods have two major limitations related to models and training data.

From the model perspective, most existing pre-trained models are not flexible enough to adapt to a wide range of vision-language tasks. On the one hand, encoder-based models such as CLIP (Radford et al., 2021) and ALBEF (Li et al., 2021) are less straightforward to directly transfer to text generation tasks. On the other hand, encoder-decoder models like SimVLM (Wang et al., 2021) have not been successfully adopted for image-text retrieval tasks.

From the data perspective, most models are pre-trained on image and alt-text pairs that are automatically collected from the web. However, these web texts often do not accurately describe the images, making them a noisy source of supervision.

2.2 Bootstrapping Language-Image Pre-training

To address these limitations, BLIP: Bootstrapping Language-Image Pre-training (Li et al., 2022) introduces two contributions, one from each perspective.

On the one hand, Multimodal mixture of Encoder-Decoder (MED) is a new model architecture that enables a wider range of downstream tasks than existing methods. An MED can operate either as a unimodal image or text encoder, or an image-grounded text encoder, or an image-grounded text decoder.

On the other hand, Captioning and Filtering (CapFilt) is a new dataset bootstrapping method for learning from noisy web data. A captioner model produces synthetic captions given web images, and a filter model removes noisy captions from both the original web texts and the synthetic texts.

BLIP achieves state-of-the-art performance on five vision-language tasks: image-text retrieval, image captioning, visual question answering, visual reasoning, and visual dialog. It also achieves state-of-the-art zero-shot performance on two video-language tasks: text-to-video retrieval and video question answering.

3 Methods

This section first introduces the new BLIP model architecture and its pre-training objectives, then explains dataset bootstrapping and finetuning architecture. Finally, testing details for the Egunean Behin dataset are explained.

3.1 Model Architecture

In order to pre-train a unified vision-language model with both understanding and generation capabilities, BLIP introduces a multimodal mixture of encoder-decoder (MED) model which can operate in three functionalities. The model architecture can be seen in Figure 1.

(1) Unimodal encoders are trained with an image-text contrastive (ITC) loss to align the image and text representations. The image encoder is a visual transformer (Dosovitskiy et al., 2020), which divides an input image into patches and encodes them as a sequence of embeddings. The text encoder is the same as BERT (Devlin et al., 2018), where a `[CLS]` token is appended to the beginning of the text input to summarize it.

(2) Image-grounded text encoder is trained with a image-text matching (ITM) loss to distinguish between positive and negative image-text pairs. It has a cross-attention layer between the self-attention layer and the feed forward layer for each transformer block. The output embedding of the `[Encode]` token is used as the multimodal representation of the image-text pair.

(3) Image-grounded text decoder is trained with a language modeling (LM) loss to generate captions for given images. It replaces the bi-directional self-attention layers in the text encoder with causal self-attention layers. A `[Decode]` token is used to as the beginning of a sequence.

In order to perform efficient pre-training and improve multi-task learning, the text encoder and text decoder share all parameters except for the self-attention layers. This is enough to capture differences between encoding and decoding tasks.

3.2 Dataset Bootstraping

As human-annotated image-text pairs are scarce, vision-language pre-training relies on large-scale image-text pairs automatically collected from the web. However, the texts often do not accurately describe the visual content of the image, making them a noisy supervision.

To address this, BLIP adds two modules, a captioner and a filter. The learning framework can be seen in Figure 2. Both the captioner and the filter are initialized from the same pre-trained MED model, and finetuned individually on the COCO (Lin et al., 2014) dataset.

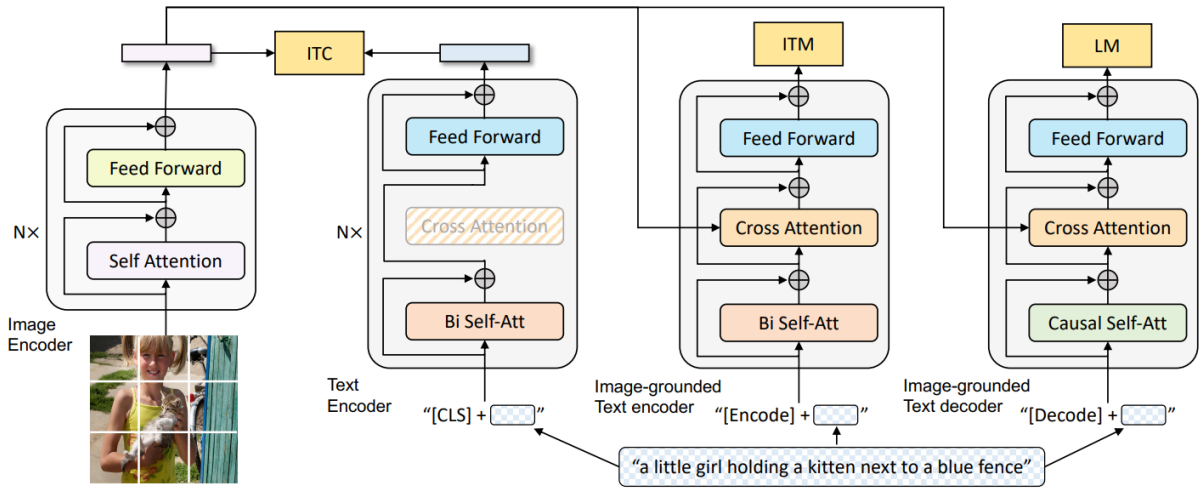

Figure 1: BLIP pre-training model architecture: multimodal mixture of encoder-decoder.

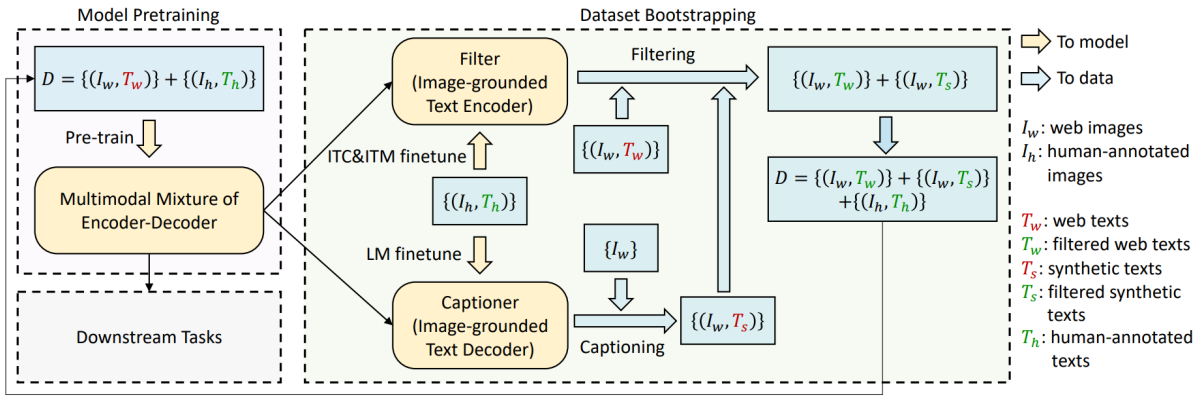

Figure 2: BLIP learning framework: a captioner to produce synthetic captions and a filter to remove noisy captions.

The captioner is an image-grounded text decoder. Given the web images, it to generates synthetic captions as additional training samples. The filter is an image-grounded text encoder. It removes noisy captions which do not match their corresponding images. Filtered image-text pairs are combined with the human-annotated pairs to form a new dataset, which is used to pre-train a new model.

3.3 Finetuning

On each downstream task, different paths of the pre-trained model are finetuned to achieve different objectives. As the task of our dataset is visual question answering, we will mainly focus on that task.

VQA (Antol et al., 2015) is a popular vision and language task. Given an image and a question about the image, the task is to provide an accurate answer. VQA[1] dataset is commonly used as a benchmark to evaluate VQA systems. Questions are generally open-ended but multiple choices are provided for some questions. Visual Genome (Krishna et al., 2017) is another popular image-text dataset that was used to finetune the model.

The finetuning architecture for VQA can be seen in Figure 3. Image Encoder, Image-grounded Question Encoder and Answer Decoder are used for this task. Encoding questions is similar to encoding the caption related to an image. Decoding an answer is similar to decoding the caption of an image.

If we compare it to image captioning, VQA requires a more detailed understanding of the image and more complex reasoning (Antol et al., 2015). The finetuning architecture for image captioning can be seen in Figure 3. The architecture is simpler, only the Image Encoder and Image-grounded Text Decoder are needed.

3.4 Testing

We test the state-of-the-art BLIP (Li et al., 2022) model on this dataset in a zero-shot setting. This

[1] https://visualqa.org

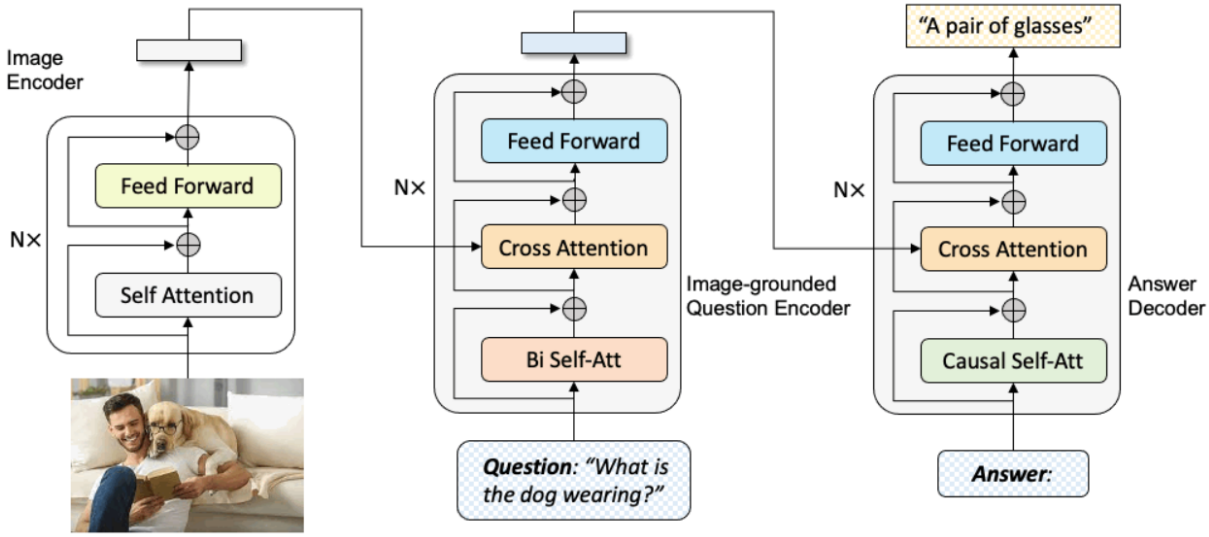

Figure 3: BLIP VQA finetuning architecture.

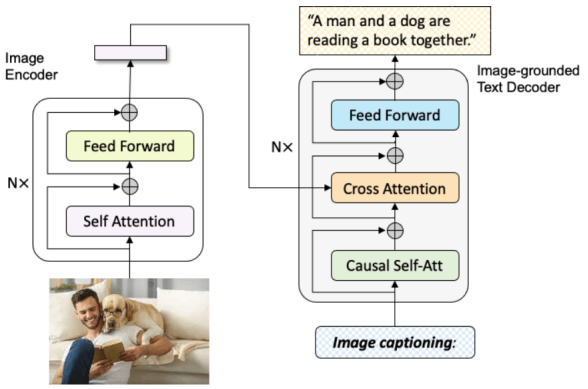

Figure 4: BLIP image captioning finetuning architecture.

will show the difficulty of each question type and the ability of the model to generalize to different types of images.

Instead of testing the model in a multiple-choice setting, we only take the most probable answer and compare it to the correct one. This makes the questions more difficult for the model. It would be interesting to also test it as multiple-choice and see the differences in results. This could be done by looking at the probabilities that the model assigns to each of the 3 answers, and selecting the one with highest probability.

Another interesting test would be creating more images and questions and finetuning BLIP model on Egunean Behin dataset. This would allow to make a comparison between the VQA dataset and Egunean Behin dataset.

4 Dataset

The aim of this dataset is to create visual questions that imitate some questions from Egunean Behin game. These can be used to evaluate models on these types of questions that require understanding visual and language content.

Three types of questions from Egunean Behin game were selected: figures, cubes and maze. There are multiple questions for each image. Questions require counting figures, colors, cubes and understanding the dimensions of the pictures.

Questions have different levels of difficulty. Figure questions only require counting objects that are visible in the image. Cube questions require more advanced reasoning about objects that are not visible. Maze questions are the most difficult ones, as the model has to understand the maze and find a path.

All the images and questions were generated automatically in two steps. First, we generate as many images as we want. Then, we generate multiple questions for each image. We decided to generate 100 images of each type to test the dataset.

The dataset and code are available at GitHub[2]. Code and data are licensed under GNU General Public License v3.0. The code is based on other works licensed under GNU GPL v3.0 that are mentioned in that repository.

[2] https://github.com/juletx/egunean-behin-vqa

4.1 Figures

Images have geometric figures of different types and colors. Figures are selected randomly to create many different images. The size of the figures and axes can be changed to create images with different dimensions and difficulty.

Images are saved with a name that contains all the necessary data to create questions. The first two digits correspond to dimension of the image. The next digits correspond to the figures in each position of the image. For example, the name of the image in Figure 5 is figures_6_4_417148_466526_041585_724774.png.

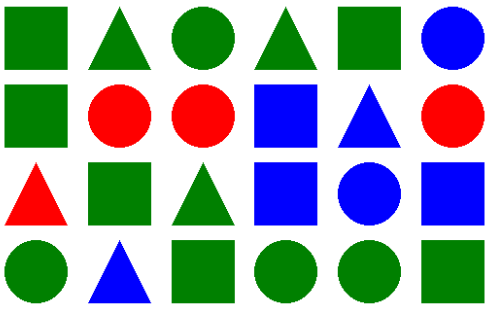

Figure 5: Figures image example.

18 questions of different types are created for each image. 3 questions about figure, column and row count. 3 questions about figure shape. 3 questions about figure color. 9 questions about figure shape and color combined. Table 1 shows example questions, correct answers (C), wrong answers (W) and BLIP answers (A) for Figure 5. Most answers of the model are wrong, but many are close to the correct answer.

Question	C	W	W	A
How many figures?	24	29	26	4
How many columns?	6	4	8	3
How many rows?	4	3	2	3
How many triangles?	6	5	8	5
How many squares?	9	7	11	9
How many circles?	9	11	8	9
How many red figures?	4	5	3	1
How many green figures?	13	15	17	3
How many blue figures?	7	9	10	1
How many red triangles?	1	3	0	2
How many green triangles?	3	4	1	3
How many blue triangles?	2	4	1	3
How many red squares?	0	1	2	1
How many green squares?	6	4	5	3
How many blue squares?	3	5	4	2
How many red circles?	3	5	4	2
How many green circles?	4	3	6	3
How many blue circles?	2	0	3	3

Table 1: Figures questions and answers that correspond to the example image.

4.2 Cubes

Images consist of 3 dimension cubes stack on top of each other. Cubes are stacked on top of each other with the defined probability. Cubes in each layer are colored with the selected colormap. The size of each axis can be controlled to create images with different dimensions.

Images are saved with a name that contains all the necessary data to create questions. The first three digits correspond to dimension of the image. The next digits correspond to the number of cubes in each position of the image. For example, the name for the image in Figure 6 is cubes_4_4_3_0002_0013_1133_3333.png.

Many questions of different types are created for each image. The number of questions for each image was 14, but it depends on the dimensions of the images. 3 questions about total, visible and non visible cubes. 4 questions about number of cubes in each x layer. 4 questions about number of cubes in each y layer. 3 questions about number of cubes in each z layer. Table 2 shows example questions and answers for Figure 6. Most answers of the model are wrong and far from the correct answers. This shows that cube questions are more difficult than figure questions.

4.3 Maze

A depth-first search algorithm is used to create mazes. The maze consist of a grid of cells and each cell initially has four walls. Starting from a given cell, the aim is to produce a path visiting each cell.

At each step we inspect the neighbouring cells. If any of them have yet to be visited, we pick one and move at random into it by removing the wall between them. If no neighbouring cell is unvisited (a dead end), then backtrack to the last cell with an unvisited neighbour.

After we construct the maze, we have to add a wall so that there is only one possible exit from our

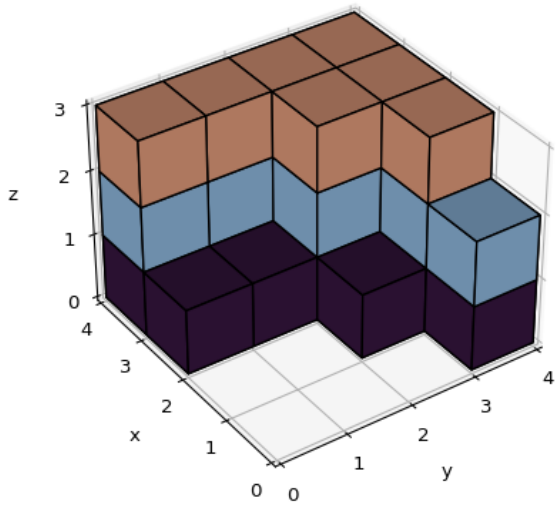

Figure 6: Cubes image example.

starting point. Each edge will have a color so that we can refer to it easier.

The size of the maze can be controlled and We can also select the starting point of the maze. Images are saved with a name that contains all the necessary data to create questions. The first digit corresponds to the index of the image. The next two digits correspond to dimension of the image. The last digits correspond to the starting and ending position. For example, the name for the image in Figure 7 is maze_0_12_8_0_2.png.

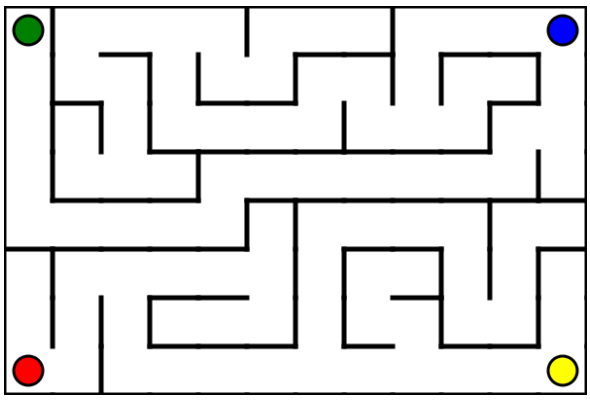

Figure 7: Maze image example.

Only 4 questions of different types are created for each image. 3 questions about cell, column and row counts. 1 question about maze exit. Table 3 shows example questions and answers for Figure 7. All the answers are wrong and very far from the correct answer. This shows that these questions are very difficult.

Question	C	W	W	A
How many cubes in total?	26	22	21	12
How many visible cubes?	17	16	11	6
How many non visible?	9	13	10	1
How many cubes in x 1?	2	0	1	6
How many cubes in x 2?	4	3	6	6
How many cubes in x 3?	8	11	10	6
How many cubes in x 4?	12	15	13	6
How many cubes in y 1?	4	6	7	4
How many cubes in y 2?	4	3	1	3
How many cubes in y 3?	7	9	8	3
How many cubes in y 4?	11	8	13	3
How many cubes in z 1?	11	10	12	6
How many cubes in z 2?	8	4	11	6
How many cubes in z 3?	7	11	10	6

Table 2: Cubes questions and answers that correspond to the example image.

Question	C	W	W	A
How many cells?	96	98	87	8
How many colums?	12	11	9	0
How many rows?	8	7	5	5
Exit from green?	blue	red	yellow	left

Table 3: Maze questions and answers that correspond to the example image.

5 Results

In this section, we compare BLIP results obtained in VQA finetuting and Egunean Behin zero-shot. BLIP also obtaines great results in many other tasks, but we only focus on the VQAv2.0 dataset. We compare the results obtained in each question type of Egunean Behin dataset.

5.1 VQA

BLIP was tested on VQA2.0 dataset (Goyal et al., 2017), which contains a total of 206k images, 1.1M questions and 11M correct answers. Images are divided for training/validation/test in sizes 83k/41k/81k. Both training and validation splits for training, and include additional training samples from Visual Genome. During inference on VQA, the decoder is used to rank the 3,128 candidate answers.

BLIP outperformes previous state-of-the-art methods on VQA as shown in Table 4. Using 14M

images, BLIP outperforms ALBEF by +1.64% on the test set. Using 129M images, BLIP achieves better performance than SimVLM, which uses 13× more pre-training data and a larger vision backbone with an additional convolution stage. Best results are achieved when using CapFilt module to filter noisy captions and generate new ones. The difference is not very big in this task, but it is clear that it has a positive impact in the results.

Method	Images	test-dev	test-std
LXMERT	180K	72.42	72.54
UNITER	4M	72.70	72.91
VL-T5/BART	180K	-	71.3
OSCAR	4M	73.16	73.44
SOHO	219K	73.25	73.47
VILLA	4M	73.59	73.67
UNIMO	5.6M	75.06	75.27
ALBEF	14M	75.84	76.04
SimVLM$_{base}$	1.8B	77.87	78.14
BLIP	14M	77.54	77.62
BLIP	129M	78.24	78.17
BLIP$_{CapFiltL}$	129M	**78.25**	**78.32**

Table 4: Comparison with state-of-the-art methods on VQA.

5.2 Egunean Behin

We decided to generate 100 images of each type to test the dataset. Each subset creates a different number of questions per image, which results in the counts that can be seen in Table 5. There are 300 images and 3600 questions, 3600 correct answers and 7200 wrong answers. Compared to VQA dataset this is a small size, but it is enough to test the performance of the model in a zero-shot setting.

Subset	Images	Questions	Acc	MAE
Figures	100	1800	16.5	3.31
Cubes	100	1400	4.79	6.79
Maze	100	400	14.25	26.39

Table 5: Number of images and questions, accuracy and mean absolute error of BLIP for each subset.

As expected, results are very bad compared to VQA dataset with finetuning. The highest accuracy is obtained in the Figures subset, which makes sense because it is the easiest one. Maze also obtains a similar accuracy, but this is due to only having four options in one of the four questions.

As accuracy is an all or nothing metric, it is necessary to calculate mean absolute error to know how close answers are to the correct one. We can see that in this case the error is bigger for the more difficult questions. In the case of figures MAE is 3.31, which means that in average answers were quite close to the correct ones. This is what we noted in the example image.

6 Conclusions

Egunean Behin is a new visual question answering dataset based on Egunean Behin game. It consists of three types of questions from the game: figures, cubes and maze. Compared to other datasets that mostly use real-world images, all the artificial images and questions are generated automatically. It can be used to evaluate VQA models in an out-of-domain zero-shot setting.

BLIP is a new VLP framework with state-of-the-art performance on a wide range of downstream vision-language tasks, including both understanding and generation tasks. BLIP pre-trains a multimodal mixture of encoder-decoder model using a dataset bootstrapped from noisy image-text pairs by creating diverse synthetic captions and removing noisy captions.

Great improvements are being made in vision-language tasks such as visual question answering. However, when evaluating in an out-of domain zero-shot setting, we can see that there is still a lot to improve. Future work should also focus on making improvements to the model and data aspects.

Regarding Egunean Behin dataset, many improvements can be made. Images of different sizes could be generated to make questions that are easier or more difficult. More questions could also be added for each image.

Adding more questions types that are present in the game would be great. For example, counting animals, finding words in an alphabet soup and completing puzzles could be added.

It would also be interesting to test BLIP as multiple-choice in this dataset and see the differences in results. Another interesting test would be to create more images and questions and to finetune BLIP model on Egunean Behin dataset. Testing other models to see the differences would also be an interesting experiment.

References

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *arXiv preprint arXiv:2201.12086*.

Junnan Li, Ramprasaath Selvaraju, Akhilesh Gotmare, Shafiq Joty, Caiming Xiong, and Steven Chu Hong Hoi. 2021. Align before fuse: Vision and language representation learning with momentum distillation. *Advances in Neural Information Processing Systems*, 34.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Zirui Wang, Jiahui Yu, Adams Wei Yu, Zihang Dai, Yulia Tsvetkov, and Yuan Cao. 2021. Simvlm: Simple visual language model pretraining with weak supervision. *arXiv preprint arXiv:2108.10904*.