Visual Question Answering

Satish Palaniappan, Ayush Agarwal, Sonakshi Grover

***Abstract*—Recent advances in deep learning have provided a platform to accomplish tasks involving multimodal learning. Visual Question Answering (VQA) is one such recent problem in computer vision and natural language processing. In VQA, an algorithm needs to answer text-based questions about images. These questions require methods to understand vision, language and common-sense domain knowledge to output an reasonable answer. In this project, we attempt to leverage transfer learning to solve the VQA problem in the domain of the popular anime series, Pokémon. In addition, we provide an insight into the domain-specific requirements for a successful transfer. Our method is able to obtain a test accuracy of 65% on some fixed basic questions applied to Pokémon battle scene images. While there is no available state of the art in the domain, we observe that we are able to match the SOTA performance on the baseline when we work with our dataset.**

I. Introduction

The emergence of many successful deep learning pipelines to facilitate multiple tasks learned end to end has led to the recent spurt of work in Visual Question Answering. The problem statement involves the development of an algorithm which can take into input an image and a textual question and outputs an answer to the question based on the visual features from the input image. Our system considers the first 150 Pokémon (Kanto Region) and some sample questions our Pokémon VQA model will try to answer are:

- Q: What Pokémon is there in the image?
  A: Pikachu.
- Q: What attack is being performed by Pikachu?
  A: Thunderbolt
- Q: What type of Pokémon is Pikachu?
  A: Electric

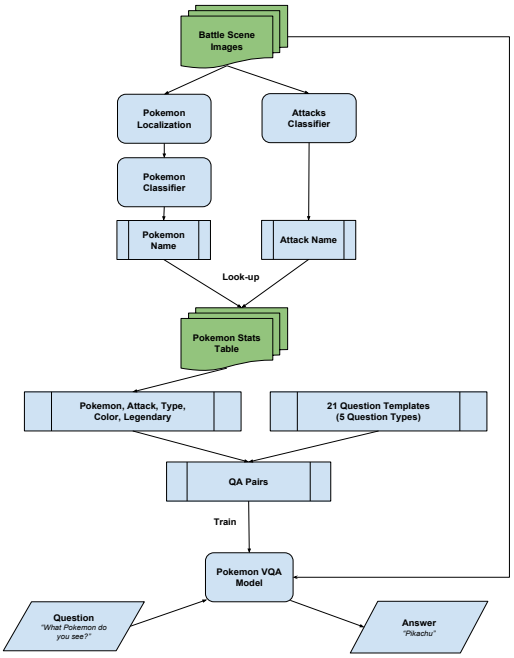
In Section 2, we start by discussing some related work in the field of VQA along with our insights we present by reproducing their results on the baseline. In section 3, we proceed with explaining the nuances of our dataset creation pipeline, where we detail the methods to construct three separate databases: Pokémons, Battle scene attacks and Questions based on Pokemon moves and statistics, which will be mapped together to generate the dataset for the VQA model. Section 4 will explore our train and transfer paradigm, where we describe the architecture and parameters of the model used to train on our dataset from scratch. Finally, we look at the results obtained on different parameters we experimented with in Section 5.

II. Related Work

We identified 3−4 implmentations of the VQA state of the art which we attempted to reproduce their results with our available computing resources. Jiang et. al. [1] introduced the Pythia which was the winning entry from Facebook AI Research (FAIR)'s A-STAR team to the VQA Challenge 2018. They make important changes to tune better hyperparameters and make a modular code for the bottom up- top down VQA model (Anderson et. al. [2]). Since each training run takes over a day, not mentioning the RAM needed to store huge Resnet-152 based features, unfortunately even with the increased compute power from the cloud credits, we were unable to reproduce the baseline from scratch. We thus moved to more optimized and smaller versions of the bottom up top down VQA model (available at [3], [4]). These involved extracting the image features from a smaller VGG-19 structure and then training an end to end network by combining the image features with textual features from the question mappings (obtained from a separate LSTM model) and applying a multi-layer perceptron on the combined features.

Note: these papers already had implementations with pre-trained weights available and hence it was merely a matter of tweaking this code a bit to get it to work on GCP and generate results on our dataset

Fig. 1. Workflow of our project pipeline



III. Dataset

The dataset creation for the Pokémon VQA problem involves the following 4 steps:

Advisor: Dr Greg Hager, Mandell Bellmore Professor of Computer Science, Johns Hopkins University.

Pokémon VQA Code: https://github.com/tpsatish95/pokemon-vqa

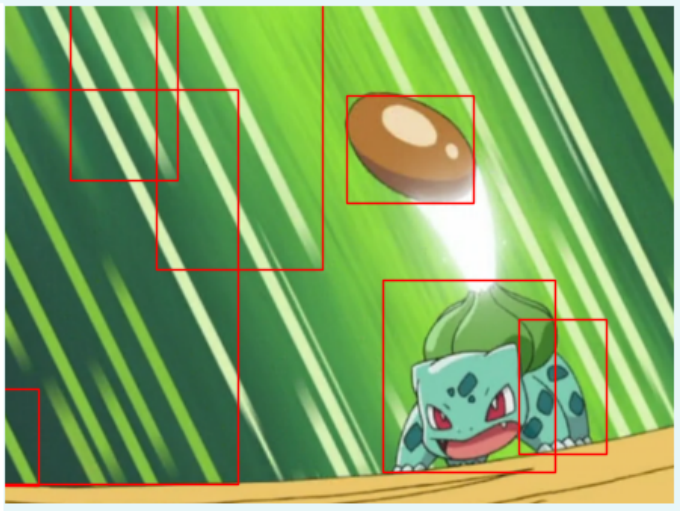
- Build a region proposal algorithm to locate Pokémon in battle scenes.
- Build a Pokémon and Attacks classifier to label the battle scenes.
- Construct questions using the Pokémon stats dataset.
- Map questions to battle scene images using the Pokémon & Attack labels.

Step 1: Build a region proposal algorithm to locate Pokémon in battle scenes:

In this step, first we used a region proposal algorithm called Selective Search[5] [6] to generate the first set of candidate region proposals. Selective search is an object detection algorithm which proposes various regions in an image that are highly probable to contain an object in them. Selective Search parameters used (best): **Scale - 350, 450, 500; Sigma - 0.8; Min. Size - 30, 60, 120, Min Area - 2000**.

But, these initial candidate region proposals are large in number, in order to reduce them quantitatively and improve them qualitatively, we came up with some Region Proposal Filtering methods like: **Merge Concentric Proposals, Contained Box Removal, Draw Super Box**. Fig. 2 shows a sample, after applying both of the above operations:

Fig. 2. Sample image obtained after Selective Search and Region Proposal Filtering



Now, from these highly probable candidates, in order to get the exact regions where the Pokémon are located. We ended up building a very simple 3 class classifier, called the Pokémon/Part/Other classifier. This used **GoogLeNet Inception V1** [7] architecture with the weights transferred from ImageNet [8]. **Accuracy: 74.9%**. The base dataset was generated by hand-labeling and had 3639 images in total with 3 classes and the dataset was balanced with 80% for train and 20% for test, which looks like Fig. 3.

Using the above-described classifier, each of the current candidate region proposals were classified. Once we know the kind of each region we have in hand, we used basic rules like merging overlapping **Pokémon** and **Part** regions while subtracting the overlapping **Other** regions from them to arrive at the final Pokémon only regions as shown in Fig. 4.

Fig. 3. Sample images from the Pokémon/Part/Other (top to bottom) Dataset

Fig. 4. Finalized region proposal locating the Pokémon

Step 2: Build a Pokémon and Attacks classifier to label the battle scenes:

Pokémon Classifier: Given that we have extracted the Pokémon only regions from the battle image, the next step would be to classify these Pokémon. We again used a **GoogLeNet Inception V1** architecture pre-trained on the ImageNet dataset for building this 150 class classifier. We pulled the base dataset for this from Kaggle [9] and Bulbapedia [10], and then augmented them by applying random image transforms such as: rotation, scaling, translation, swirl, shear, horizontal/veritical flips, and blur. Then this augmented dataset was superimposed on typical battle scene backgrounds like the ones found here. This dataset had a total of 134508 images and 150 classes, which were generated from just 1076 base Pokémon images. These images were then augmented using the techniques mentioned above to 5x, which is 5380 images. Then these augmented images were imposed on 25 distinct background images to get the final 134508 images, with a 70-30 train-test split. See dataset sample Fig. 5.

This model achieved a Top-1 accuracy of **91.16%** and Top-5 accuracy of **98.03%** with Backgrounded and Augmented dataset, while it achieves only Top-1 accuracy of **85.39%** and Top-5 accuracy of **96.43%** with just the Augmented dataset.

Fig. 5. Pokémon Classifier Dataset Sample

The Hyperparameter configuration used are shown in Fig 6. Some sample results of the classifier are shown in Fig 7.

Fig. 6. GoogLeNet Inception V1 Best Hyper Paramter Configuration

```
net: "googlenet/train_val.prototxt"
test_iter: 1407
test_interval: 4000
test_initialization: false
display: 40
base_lr: 0.001
lr_policy: "step"
stepsize: 32000
gamma: 0.1
max_iter: 10000000
momentum: 0.9
weight_decay: 0.0002
snapshot: 4000
snapshot_prefix: "bvlc_googlenet_pokenet"
solver_mode: GPU
```

Attacks classifier: For this classifier we will again be using the **GoogLeNet Inception V1** architecture, but this time around, we do not need the Pokémon regions as attack classification is much similar to a scene classification problem than an object classification problem. So, we used a GoogLeNet pretrained on the **MIT Places 205**[13] dataset to transfer learn this attacks classifier. The attacks dataset was crawled from Bulbapedia [10] and Pokémon Wikia[14]. Which was then augmented using similar techniques as discussed above to get 21188 images with 144 attack classes finally, which was split into 70-30 for train-test. Some attack samples (Thunderbolt, Flamethrower, Water Gun, etc) can be found in Fig. 8, there are 144 attacks in total.

We obtained a **88% top-5 accuracy and a 55% top-1 accuracy** for this Attacks Classifier. Though these accuracies are low, using restricted Pokemon-Attacks mappings we boosted

Fig. 7. Pokémon Classifier Top-5 results

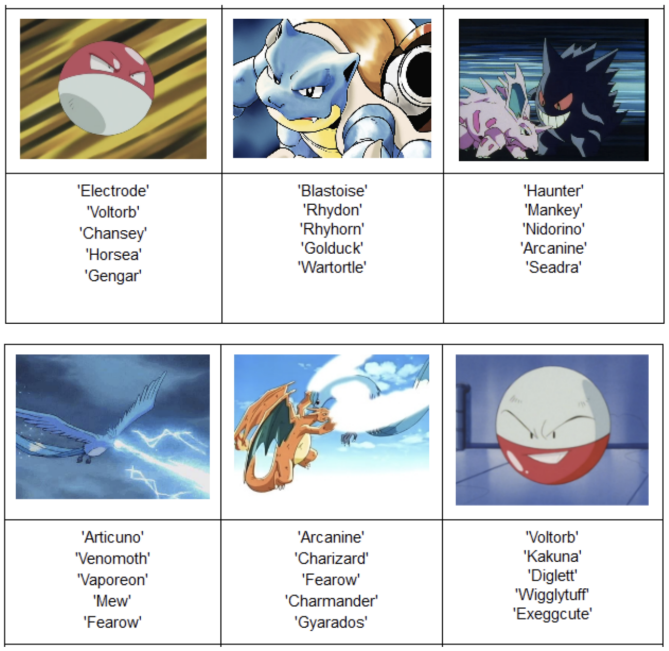
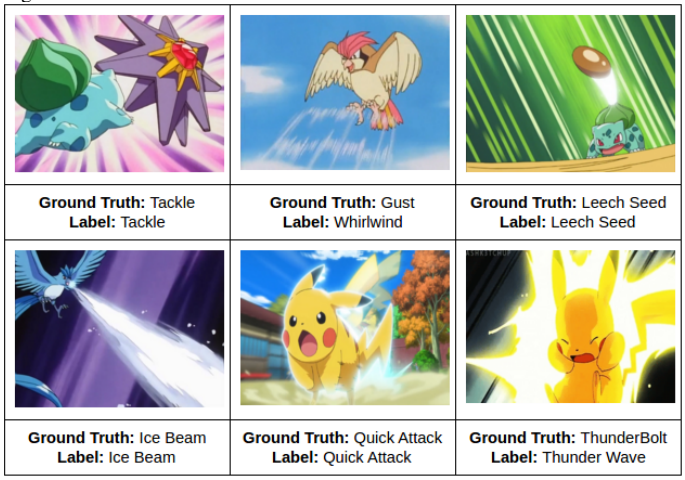


Fig. 8. Attacks Classifier Dataset

the accuracy of these classsifiers based on the Pokémon data we figured out. Because a given Pokémon can only perform 10 attacks and not all 144 of them. Fig 9. Shows some sample attack classifier results.

Fig. 9. Attacks Classifier Results



Step 3: Construct questions using the Pokémon stats dataset:

Before we see how we gathered data to construct questions,

lets see how we built the battle scenes dataset, which will essentially be the image input to our VQA model. We crawled Bulbapedia for battle scene images and got 1511 images in total which were split into 80-20 for train-test. Once we got these input VQA images, we then feed these into the Pokémon and Attacks Classifier designed in Step 2, to get the Pokémon present and Attack being performed information for a battle scene. Then we use this to lookup the Pokémon Stats [12] table and get more information like the Pokémon's type, color and legendary information for generating more QA pairs as described in Step 4.

Step 4: Map questions to battle scene images using the Pokémon & Attack labels:
Once we process the input battle scene images as in Step 3, we will have sets of [Battle Scene Image, Pokémon, Attack, Color, Type, Is Legendary] for each of the 1511 images. Then we generated 21 questions for each of these images with 5 distinct types (as listed below) and other 16 questions were equivalent/paraphrased versions of these:

1) What pokemon is there in the image?
2) What attack is being performed by the pokemon?
3) What type of pokemon is it?
4) Is the pokemon legendary?
5) What is the color of the pokemon in the image?

So, we got the Pokémon and attack names (from step 2), Pokémon color, type and legendary data (from step 3) and then basically now are doing a cross product of all 21 questions for each of the 1511 battle scene images, to generate the 31,731 QA pairs with a 70-30 split for train-test.

Before feeding these QA pairs to the network we preprocess it. We tokenize the sentences and map each token(word) to an id. The vocabulary size is 37 words. We have 304 possible answers including the different attacks, colors and Pokémons. After this, we create a JSON object for every QA pair wherein the keys of JSON object comprises of question ID, image path, question and answer.

IV. Model Architecture and Training

We finally used the model available at [15]. The model structure involves looking at the image input, reads a question and comes up with an answer and an attention heatmap denoting the most important regions in the image it observed to output the answer. The model also supports referring the image repeatedly with intermediate attention maps before producing the final answer.

Compared to models that simply combine the question vector and the global image vector, attention models construct a more informative query vector at each stage since higher weights are put on the visual regions that are more relevant to the question. Each attention layer extracts more fine-grained visual attention information for answer prediction.

For the kth attention layer, the following is the formulation of these stacked attention networks:

$$h_A^k = tanh(W_{i,A}^k * v_i + (W_{Q,A}^k * u^{k-1} + b_a^k)) \quad (1)$$

$$p_i^k = softmax(W_p^k * h_A^k) \quad (2)$$

The first equation is the activation generated on combining the image features obtained from the VGG 19 network and the question vector. The second equation applies a softmax function to generate the attention map over the image. Then the aggregated image feature vector is added to the previous query vector to form a new query vector:

$$v'^k_i = \sum_i p_i^k * v_i \quad (3)$$

$$u^k = v'^k_i + u^{k-1} \quad (4)$$

This vector is propagated through the network until the final answer is produced. We tried hyperparameter tuning, grid searching through them and chose to use the following parameters as they worked best for our dataset.

Fig. 10. Diagram summary of the model from the paper

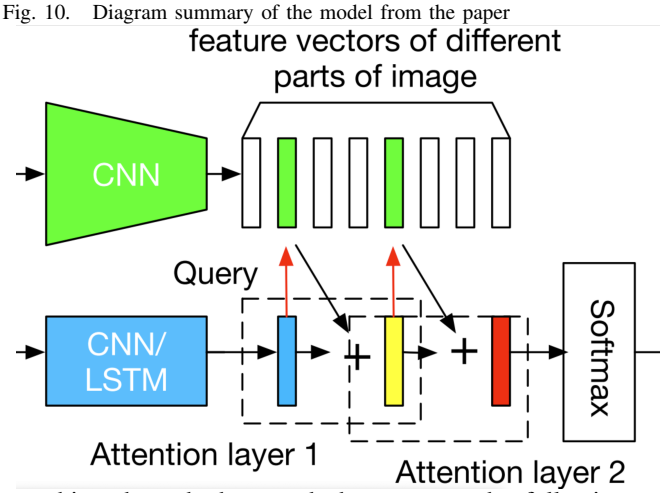


TABLE I
Model Parameters

Batch Size	256
Learning Rate	0.001
Number of Iterations	800
Number of Attention Layers	2
Word Embedding Size	200

V. Results

In order to evaluate our VQA Model we used the battle scenes dataset we built, which has 1511 battle scene images, with 31,731 QA pairs. This was split into 70-30 for train-test. When we applied our best VQA Model with model parameters as described in *Section IV*, we got an **accuracy of 65.9%** after 800 iterations. This accuracy was calculated with strict equivalence of answers and no points were given if the answer was even nearly correct, making our accuracy score strict and even a low accuracy score means, better results than the normal VQA Models. Refer Fig. 11 for the some of the best results our VQA Model produced, and Fig. 12 for some of the best and not-so-perfect yet meaningful results our model produced. Our model outputs completely meaningless results only in very rare cases.

Fig. 13 plots the change in Training loss with the number of iterations the network is trained. We see that as the training progresses, the curve undergoes a smooth decrease finally saturating at 800 iterations. Hence our model learns well.

Fig. 11. Best VQA Results (All Perfect)

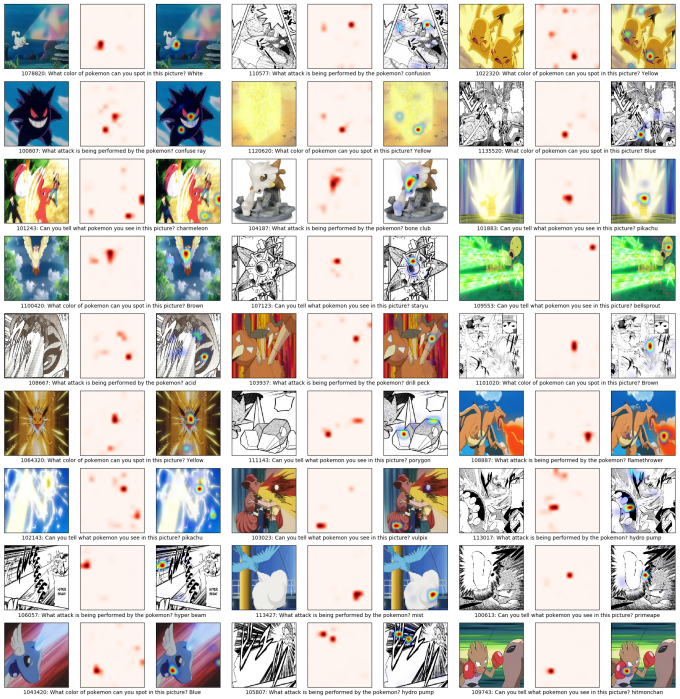
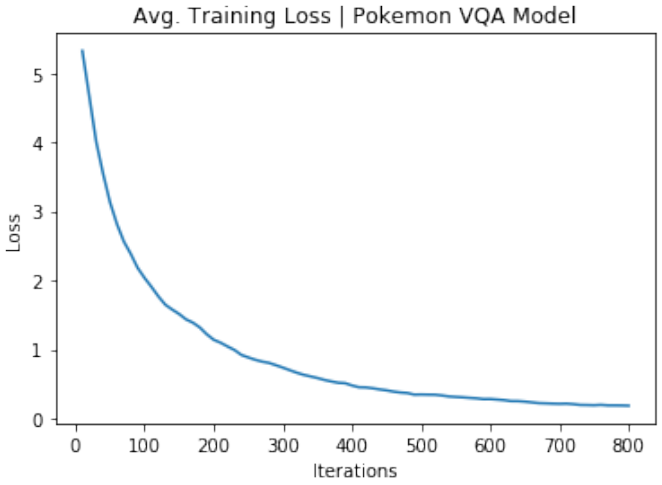


Fig. 12. Best VQA Results (Not so Perfect yet Meaningful)

From the results obtained on baseline datasets (as discussed in [15]), we can see that the performance of the VQA network on our dataset is good. VQA network gave an accuracy of 63.82% on MSCOCO which is comparable to our 65.9% In fact the network performs surprisingly well on the Pokémon dataset taking into account that our images are very bright, filled with fire/water/grass type Pokemons which are scattered across the images and are not localized. Also unlike MSCOCO, our dataset has lesser number of discriminative



objects in its images. Hence intuitively, we had perceived this to be a harder problem as compared to benchmark datasets, but the VQA model outperformed our expectations.

On analyzing some specific results we get a sense of how robust the model is. For instance, let us consider questions 108887 (6th Row, 3rd Column, Fig. 11) and 113427 (8th Row, 2nd Column, Fig. 11), both of them are ”What Attack?” questions and the model perfectly predicts ”Flamethrower” and ”Mist” respectively. What makes it more interesting is the fact that, it attends to/focuses at the correct place in the image to arrive at this answer. Proving that our answers are not just random and that the model makes sensible decisions. For example, in the first case it attends to the fire breath of Charizard and in the second case it attends to the cloud of mist from Articuno’s mouth to answer the questions. The next interesting case is question 102143 (7th Row, 1st Column, Fig. 11), where we ask the model, what Pokémon it sees in the image, it perfectly focuses on the very small part of the image where is sees the Pikachu which is almost fully occluded by the Thuderbolt it produced and suggests that it was Pikachu, which is really interesting. Another question which really surprised us, was question 1135520 (2nd Row, 3rd Column, Fig. 11), where we have black and white image of Articuno performing an attack and the question was a ”What Pokemon Color?” type question and to our surprise the model predicted perfectly that the color was Blue, which means that model first figured our that the Pokémon was Articuno and then it inferred from this fact that the color was Blue! Let us analyze one last image where the model predicted wrong, but did predicted a nearly perfect answer instead. See question 1020416 (2nd Row, 2nd Column, Fig. 12), where the question is ”Is the Pokémon legendary?” and as the model does not know the correct answer for this question with high confidence it predicts Pikachu as the output, which is actually the answer to the ”What Pokémon?” question, which is really sensible.

Hence, the Pokémon VQA network is so robust and universal that it not only performs well to a specific dataset (MSCOCO) but also performs exceedingly well on our Pokémon dataset whose images are so obstructed and random. This is a huge step forward in research of VQA as we are

literally able to answer questions about all kinds of images, thus helping us answer a broad spectrum of images.

VI. Discussion

As can be seen above, we received positive results for the tough domain transfer in the VQA problem. We learned that VQA helps us to answer questions pertaining to all kinds of images, making it a highly useful tool. As a future problem, with additional compute resources and time available, we would like to attempt the transfer for Pokémon scene images having multiple pokémons and even trainers present in the image. As of now we had the another limitation of constraining our number of questions due to less visual content in the image. One of the advanced topics we look forward to working on involves predicting the outcome of a battle by looking at the battle stage in the image and some additional prior about the Pokémon weaknesses and strengths.

Also, due to the less visual content in our images, we could only form questions having a fixed answer, though in real world (in the pokeḿon world too!) there can be multiple correct answers to a single question. For that we came across [16] which deals with automatically predicting from a visual question whether a crowd would agree on one answer or not. It also proposes how to exploit these predictions to efficiently collect all valid answers to visual questions.

Through this course we learned to leverage the power of deep learning models to solve problems on large datasets efficiently and accurately. We also learned to reason about why our changes to the models work and the possible intuition of the mathematics behind it. As experienced deep learning students, we would now imagine looking at every paper or model we come across with a mathematical and logical bend about their working.

We highly recommend this course to all students as it introduced us to all the current Deep Learning techniques being used in the real world. We believe that knowledge of these techniques is crucial to our understand of how everything that surrounds us is being predicted/classified, in this era of digitization. For a student, taking this course would not only mean that you need to understand the mathematics behind the algorithms involved but would also require you to experiment with the parameters involved in these algorithms, in order to understand how these parameters are controlling the behaviour of the algorithms. The course will keep you busy in optimising your results as well as analysing why you are getting a particular result. We appreciate the efforts put in by the instructors including the professor and the TAs.

References

[1] Jiang, Yu, Vivek Natarajan, Xinlei Chen, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. "Pythia v0. 1: The winning entry to the vqa challenge 2018." arXiv preprint arXiv:1807.09956 (2018).

[2] Anderson, Peter, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. "Bottom-up and top-down attention for image captioning and visual question answering." In CVPR, vol. 3, no. 5, p. 6. 2018.

[3] https://github.com/iamaaditya/VQA_Keras

[4] https://github.com/anantzoid/VQA-Keras-Visual-Question-Answering

[5] Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T. et al. Int J Comput Vis (2013) 104: 154. https://doi.org/10.1007/s11263-013-0620-5

[6] https://github.com/AlpacaDB/selectivesearch

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1-9

[8] http://www.image-net.org/

[9] https://www.kaggle.com/thedagger/pokemon-generation-one/discussion

[10] https://archives.bulbagarden.net/wiki/Special:ListFiles

[11] https://www.kaggle.com/n2cholas/competitive-pokemon-dataset

[12] https://www.kaggle.com/rounakbanik/pokemon

[13] http://places.csail.mit.edu/downloadCNN.html

[14] http://pokemon.wikia.com/wiki/Pok%C3%A9mon_Wiki

[15] https://github.com/abhshkdz/neural-vqa-attention

[16] Danna Gurari and Kristen Grauman. 2017. CrowdVerge: Predicting If People Will Agree on the Answer to a Visual Question. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). ACM, New York, NY, USA, 3511-3522. DOI: https://doi.org/10.1145/3025453.3025781