COMPARATIVE ANALYSIS OF CNN AND FASTER R-CNN IN VISUAL QUESTION ANSWERING: EXPLORING OBJECT DETECTION EFFICACY

TECHNICAL REPORT

Michael Nørbo
Department of Electrical and Computer Engineering
Aarhus University
Finlandsgade 22, 8200 Aarhus
202202966@post.au.dk

Dilan Celebi
Department of Electrical and Computer Engineering
Aarhus University
Finlandsgade 22, 8200 Aarhus
202202967@post.au.dk

December 8, 2023

ABSTRACT

This study examines Faster R-CNN and conventional CNN models in Visual Question Answering (VQA) on the VQA v2.0 dataset. It explores if Faster R-CNN's object detection enhances VQA performance. The report presents findings on overfitting, dataset adequacy, and sampling biases, indicating that performance is curtailed by limited data and computational power, pointing to the need for more extensive data in future work.

Keywords VQA · Faster R-CNN · CNN · Data Sampling

1 Introduction

In the field of Visual Question Answering (VQA), the integration of advanced image processing techniques significantly enhances a systems capability to interpret and respond to complex queries. This study explores the efficiency of employing Faster R-CNN, an advanced object detection model, in contrast to conventional CNNs for VQA tasks. Our focus is to evaluate whether the enhanced object detection and segmentation abilities of Faster R-CNN can provide a more nuanced and context-aware analysis, potentially elevating the overall performance of a VQA model.

2 Related Work

The paper "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering"[1] marked a paradigm shift in the VQA field by integrating Faster R-CNN with attention mechanisms. This innovative approach, combining detailed object detection (bottom-up attention) with contextual question understanding (top-down attention), significantly improved VQA performance compared to existing models.

Examining the current (Dec 2023) state-of-the-art models[2], different approaches are used in the 3 top-scoring models on accuracy. Notably, the VLMO model[3] employs a Mixture-of-Modality-Experts Transformer, allowing for a more flexible and efficient handling of vision-language tasks, but does not specifically utilize Faster R-CNN. Another significant approach is the BEiT-3 model[4], which employs Multiway Transformers for a unified handling of images, texts, and image-text pairs, diverging from the typical use of CNNs or R-CNNs for image processing. Additionally, the PaLI model[5] combines large-scale language and vision components in a unified framework, using Transformer-based architecture for both modalities, which is a step away from the conventional use of CNNs or R-CNNs. These studies illustrate the evolving landscape of VQA and the increasing emphasis on multimodal, integrated approaches that our research aims to contribute to. A study by Agrawa et al.[6] also emphasizes the importance of large-scale datasets and

diverse question-answer pairs in training robust VQA models. It is also discussed the challenges in creating models that can interpret and understand a wide range of questions, from simple object identification to complex queries requiring deeper reasoning.

Inspired by Anderson et al.[1] and their revolutionaly concept from 2017, our project investigates the standalone impact of Faster R-CNN over conventional CNNs in VQA tasks. The hypothesis is that Faster R-CNN's ability to detect and segment objects within images will allow for a more detailed and context-aware analysis, potentially improving the VQA system's ability to understand and accurately respond to a broader range of questions. By exploring the differences between CNN and Faster R-CNN, knowledge can be gained to provide fundamental insights into how object detection models influence VQA systems, which could provide contextual relevance in more advanced models (e.g. state-of-the-art approaches).

3 Method

The code is available on: `https://github.com/heltechael/VQA_cnn_versus_rcnn`, and the dataset can be downloaded through: `https://visualqa.org/download.html`.

3.1 Dataset Analysis

The dataset utilized in this project is the Visual Question Answering v2.0 (VQA v2.0)[7] dataset, which is designed to challenge and assess the synergy between image comprehension and language processing. The dataset contains 82,783 COCO images along with a minimum of three questions (averaging 5.4) per image. In total, there are 443,757 questions and 4,437,570 answers. Each question is accompanied by ten ground-truth answers and three plausible (but likely incorrect) answers as distractors. VQA v2.0 is a formidable testbed for VQA-related models, and is the top dataset on `paperswithcode.com` for Visual Question Answering.

Illustrating and inspecting the distribution of answers in the VQA v2 dataset, a significant imbalance in the distribution of answer types was observed:

- yes/no: 37.61%
- numbers: 12.96%
- other: 49.41%

The majority of answers are 'yes' or 'no', taking up 37.61% of the total answers, which indicates a prevalence of binary questions. Additionally, numerical answers such as (e.g. '1', '2', '3') account for 12.98% of the answers, making $\sim 50\%$ of the answers either binary yes/no or a number. Among the other answers are colors (e.g. 'white', 'blue', 'red'), as seen in the most common answer types, which suggests a variety of questions related to object identification by color as well. The 10 most common answers are seen below in table 1.

'yes'	**'no'**	**'1'**	**'2'**	**'white'**	**'3'**	**'blue'**	**'red'**	**'black'**	**'0'**
84978	82516	12540	12215	8916	6536	5455	5201	5066	4977

Table 1: Frequency of the most common answers

Examining the distribution of the most common question-types shows that the dataset appears to be skewed towards 'what' questions with over 175000 questions of this type, which are significantly more frequent than other types. The next-most represented question type is 'how many' at around 50000 questions, followed by 'is there' questions at $\sim$20000, 'where' $\sim$15000, and 'which' at $\sim$7000. This imbalance could affect the model's performance, potentially making it better at answering 'what' questions compared to others like 'where', 'how many', or 'which'.

The dataset has a total of 22,531 unique answers. Since the dataset contains a wide spectrum of possible responses, it is clear, that the model needs to understand and categorize a broad array of visual data. The average question length is 6.2 words, which suggests that questions are typically concise, likely targeting specific details within images.

Inspecting a sample of datasets questions reveal a wide range of inquiry types, from simple identification, such as "What type of expression does the woman have?", to more location and quantity-specific questions like "Where is the man's hand at?" and "How many animals are there?". These examples highlight the datasets complexity and the necessity for a model capable of understanding and answering a broad spectrum of question formats.

3.2 Network Achitecture

The architecture for the model is designed with the task in mind: comparing a baseline CNN approach to an approach utilizing Faster R-CNN. Therefore, two models are proposed:

- A baseline CNN model
- A model leveraging Faster R-CNN backbone

A high abstraction depiction of the model is shown in figure 1. The model is implemented in Python using PyTorch[8]. For comparison between the two ways to extract image features, it is essential that the models are similar in all other aspects.

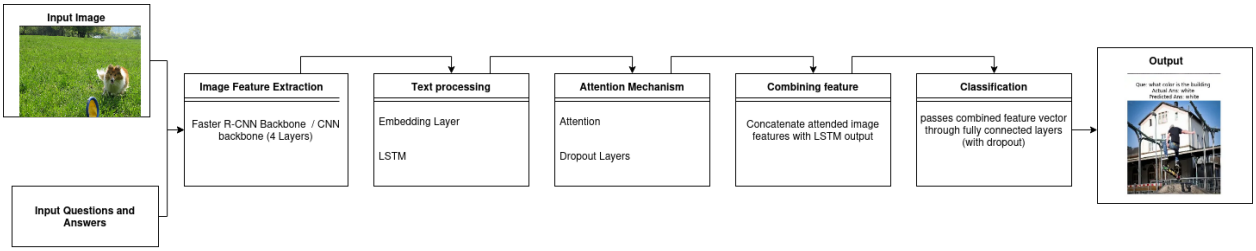

Figure 1: High abstraction depiction of the model architecture

3.2.1 Image Feature Extraction

The image feature extraction in both models is crucial for visual understanding. The baseline CNN model utilizes a ResNet-50 model (`resnet50` from `torchvision.models`)with weights that are pre-trained on ImageNet, taking advantage of transfer learning. Since ImageNet contains millions of images that are similar to the images in the VQA v2.0 dataset, the idea is that the pre-trained model has learned a rich representation of features from a vast amount of data, which can be transferred to this problem. All layers of ResNet-50 are kept except the last two layers, which leaves only the convolutional layers of the network, thereby acting as feature extractors. Therefore, the truncated model outputs a feature map instead of a final classification score, which is the starting point for further training.

In the Faster R-CNN model, the image feature extraction is achieved through the utilization of the `fasterrcnn_resnet50_fpn` model from `torchvision.models`, which is also pre-trained on ResNet-50 as a model for object detection. The model utilizes the backbone (`fasterrcnn.backbone.body`) without the RPN and ROI heads to only extract feature maps from the input images. An adaptive pooling layer is used to capture global features from the entire image, which is essential in VQA for understanding the full context of the image, and also to standardize the size of the feature maps.

3.2.2 Text Processing

To process the natural language questions into a format the model can process, a text processing component is necessary.

The model uses an embedding layer (`nn.Embedding`), which maps each word in the question to a high-dimensional vector space. This layer was instantialized with a vocobulary size equal to the number of unique words in the dataset (`vocab_size`) and an embedding size (`embed_size`). These vectors were learned during trained to represent the semantic meaning of words.

Following the embedding layer, a Long Short-Term Memory (LSTM) network (`nn.LSTM` from `torch.nn`) was used. This layer was designed to process sequences of data (in this case: sequences of word embeddings) with the intent of capturing long-term dependencies and contextual information. The LSTM processes the embedded questions and outputs a feature representation of the entire question.

Finally, a dropout layer (`nn.Dropout`) was followed the LSTM to better generalization of the model and reduce overfitting. This layer randomly zeroes some of the elements of the input tensor with a probability, $p = 0.5$, during training to combat overfitting. The output of the text processing was a dense representation of the question, which can then be used with the image features to process further.

3.2.3 Attention Mechanism

To integrate image and text features effectively, an attention mechanism was used. This enables the model to dynamically focus on different parts of the image depending on the question's context, which should enhance the models ability to accurately answer a wide range of queries.

Using linear transformation the features from both the image- and text-domain were brought into a common space where they can be compared and combined. Two linear linears (`image_att` and `text_att`) are separate (fully connected) linear layers that transform the image and text features into a common hidden dimension. The transformed features were combined using addition and ReLU activation to generate a unified representation. This allows the model to consider both visual and textual elements when focusing on specific parts of an image. An attention score was calculated using another linear layer (`final_att`) along with softmax activation, which outputs the relevance of each image region in regards to the given question. The image features were then weighted by the attention scores, thereby emphasizing the areas more relevant to the question.

3.2.4 Classification

To predict an answer, the model passes the combined image and text features through two fully connected (linear) layers. The first layer (`fc1`) reduces dimensionality of the combined feature to a smaller size (512), which was chosen as a common practice to compact the information before the final classification. ReLU activation functions are applied to introduce non-linearity, which enables the model to learn complex patterns, and dropout is used again. The second and final classification layer (`fc2`) maps the reduced features to the number of possible answers (`num_answers`)). The output of this layer represents the model's predictions for each possible answer. The final output can then be processed with softmax during training to obtain the probability distribution over possible answers.

3.3 Data Pre-processing

In the data pre-processing, several steps were carried out to prepare the dataset for training and evaluation.

3.3.1 Sampling and Tokenizer

Since the dataset is very large, and computational power is limited, measures were put in place to reduce the complexity. Instead of using the entire dataset, a random subset of n data points from the training set was chosen to create a manageable training dataset. Similarly a subset of k samples are chosen for the validation and testing sets. This sampling process ensured that the dataset remained representative while downsizing for computational efficiency.

When sampling a subset of the full dataset, it is crucial to ensure that the validation and testing set does not contain unseen answer labels (labels not contained in the training set). Therefore, a unique list of answer classes are computed for the training set, which is used to filter the validation and test sets by excluding samples with unseen answer classes.

The questions were tokenized using a custom tokenizer. This tokenizer was fit on the combined set of questions from training, validation, and test sets to ensure a consistent vocabulary. Each question was then converted into a sequence of integers, where each integer represented a specific word in the tokenizer's vocabulary. To handle variable lengths of questions, padding was applied to standardize the length of all question sequences to the maximum question length observed in the dataset.

For answers, label encoding was utilized to convert text-based answers into numerical classes. This process involved fitting a label encoder on the answers from the training dataset, which assigned a unique integer to each unique answer. The same encoder was then applied to the answers in the validation dataset to ensure consistency in encoding across datasets.

3.3.2 Image Processing (augmentation)

A series of transformations was applied to standardize and augment the image data. The raw images are of sizes around 640x480 (or reversed), but due to computational limitations, all images were resized to a fixed dimension of 224x224 pixels. Additionally, 224x224 is the minimum height and width for utilizing PyTorch models[9].

Data augmentation was used for the training data to introduce variability and improve the robustness of the model. The augmentation includes random horizontal flipping with a 50% probability, random rotation by up to 15 degrees, and adjustments to contrast, saturation, and brightness. The images were converted to tensors and normalized using predefined mean and standard deviation values from the ImageNet dataset, which the VQA v2.0 dataset uses: mean $= [0.485, 0.456, 0.406]$ and standard deviation std $= [0.229, 0.224, 0.225]$.

3.3.3 Dataset and DataLoader Configuration

Custom dataset classes were defined for both the training, validation, and testing sets, which took the pre-processed images, questions, and answers as input. PyTorch's DataLoader was then used to batch and shuffle the training data, which makes loading and processing the data efficient during training. The validation data was also batched but not shuffled, as the order of the data does not affect model evaluation. This setup ensured efficient and effective feeding of data into the model for both training and evaluation phases.

3.4 Training Strategy

The training strategy for our Visual Question Answering model faced significant computational challenges. Training the models on the a 15000 training sample subset with 3000 validation samples cost 7.5 compute units per hour using a V100 or T4 GPU. Therefore, our approach involved training and evaluating the model on a small subset of the data (1,000 training and 500 validation samples) to refine hyper-parameters to manage these constraints at the best of our abilities.

For training setup, the VQA model (with either CNN or Faster R-CNN backbone) was initialized with specific parameters, including learning rate, weight decay, and Adam optimizer. A cross-entropy loss function was chosen for training, because it is effective at handling multiple classes in classification tasks. A learning rate scheduler (`ReduceLROnPlateau`) was used to adjust the learning rate based on validation loss to enhance the models ability to converge to a minimum loss and avoid overfitting when a metric stopped improving. Early stopping was implemented to stop training if improvement is observed over a specified number of epochs. Gradient clipping were experimented with, but not included in the final training setup.

After fine-tuning the hyper-parameters on a smaller subset, especially focusing on mitigating overfitting, the parameter values chosen for testing was the following. A high epoch count of 1000, with early stopping set at 5 epochs of no validation loss improvement. This ensured the model trained until convergence. The initial learning rate was set at 0.001, optimized for performance in initial tests. To address potential overfitting, a weight decay of 0.001 was applied to introduce L2 regularization. This approach encourages the model to learn smaller weights, promoting simpler models to prevent overfitting. The learning rate scheduler, with a factor of 0.1 and a patience of 3 epochs, was utilized to adjust the learning rate based on the validation loss. Seperately, a the decay factor of 0.999 was separately applied to gradually reduce the learning rate over time.

The training was designed as a training/validation split, which means that for every epoch, the model trains on the training set and validates on the validation set. This outputs two losses: a training loss and a validation loss. Since the goal of the training is to improve the models ability to generalize on unseen data, adjusting training parameters based on the validation loss helps in assessing the models performance. This enables more informed decisions when tuning hyper parameters, as it is more transparent how the model adjusts and learns during training.

The training process spans multiple epochs, incorporating learning rate decay to stabilize training. Each epoch consists of a training phase, where the model learns from the training data, and a validation phase for performance assessment. The best model state is saved based on the lowest validation loss, and the early stopping mechanism will stop the training if no improvement was observed over 5 number of epochs. Training performance is logged for each epoch. Upon stopping training, the best model will be loaded, as this is concluded the best model state during training.

3.4.1 Evaluation Metrics

For evaluation, the model is assessed using accuracy, precision, recall, and F1 score on the testing set. The testing set is completely unseen, and has not been involved in the training or tuning of the hyper-parameters. Therefore, evaluating on this set depicts the models ability to generalize to unseen data.

Evaluation on the test set includes categorizing answers (e.g., yes/no, numbers, others) and computing category-wise performance metrics. This is inspired by the *VQA Challenge*[10], which is organized by the VQA team, where model entries are evaluated based on the three categories: 'yes/no', 'numbers', and 'others'. Additionally, a selected number of samples from the test set are visualized to demonstrate the model's predictions in context with the input images and questions. This evaluation approach helps in understanding the models strengths and areas for improvement across different question types and answer categories.

4 Results

4.1 Training and Validation Performance

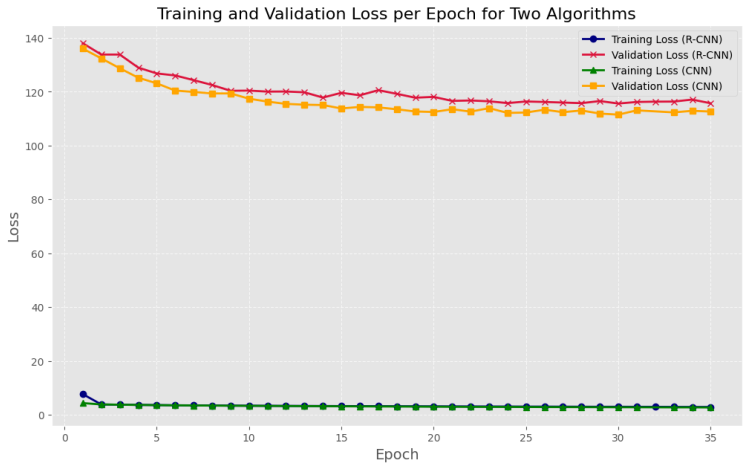

Figure 2: Learning Curves

The training and validation loss performance are analyzed over multiple epochs. The R-CNN algorithm exhibits a consistent and steady decrease in validation loss across epochs. The training loss for the R-CNN algorithm shows a consistent and gradual decrease over the course of multiple epochs until the 5th epoch. After the 5th epoch, the training loss stabilizes, indicating that the model has effectively learned. In contrast to R-CNN, the CNN algorithm demonstrates more fluctuation in both validation and training losses.

4.2 Testing Performance

Table 2: R-CNN Test Results

Category	Accuracy	Precision	Recall	F1 Score
yes/no	0.5072	0.5373	0.5072	0.4148
numbers	0.2299	0.0684	0.5220	0.2178
others	0.1190	0.0447	0.3341	0.1550
overall	0.2953	0.2426	0.4593	0.3229

Table 3: CNN Test Results

Category	Accuracy	Precision	Recall	F1 Score
yes/no	0.5071	0.2603	0.5071	0.3440
numbers	0.2278	0.2370	0.3083	0.2062
others	0.1300	0.0803	0.2779	0.1754
overall	0.3038	0.1694	0.4147	0.2752

5 Discussion

The performance of two models is compared across various metrics. The R-CNN model demonstrates balanced results across each category, with the highest accuracy observed in the 'yes/no' category at 50.72% and an overall accuracy of 29.53%. In contrast, the baseline model, while achieving lower results in the 'yes/no' category, exhibits slightly superior overall performance with an accuracy of 30.38%. Comparing the baseline CNN model and the R-CNN model across different categories shows that both models have similar accuracy for 'yes/no' questions, but the R-CNN model has a higher precision and F1 score. For 'numbers', the R-CNN model has a lower precision but significantly higher recall, resulting in a higher F1 score. In the 'others' category, both models struggle, but the R-CNN again has a higher

recall. Overall, the R-CNN model shows improved recall but slightly reduced accuracy, indicating a better ability to identify relevant answers but also a higher chance of false positives. The F1 scores suggest that the R-CNN may be more balanced overall in terms of precision and recall, despite the slightly lower overall accuracy.

A concerning observation was made when analyzing the accuracy of the 'yes/no' category, which is the category with the highest accuracy for both models at $\sim 50\%$. This metric is notably concerning because it implies that the models are performing on par with *random guessing*, which would also be 50%. This observation suggests that the models have not learned to effectively discern 'yes' or 'no' answers, which should intuitively be one of the easier categories due to its binary nature. Therefore, the 'yes/no' category does not show superior performance compared to more complex categories, raising concerns about the models binary classification capabilities.

For 'numbers', there is a significant variation between precision and recall, especially in the R-CNN model. The high recall and low precision indicate that while the model can identify relevant cases, it also incorrectly labels many non-relevant cases as relevant. The 'others' category has the lowest accuracy, which is expected as it is the most diverse and challenging category. However, the low precision rates suggest that the models are highly unreliable in their predictions for this category. The overall F1 scores being below 0.35 for both models indicate that neither model is particularly effective at balancing precision and recall across all categories.

These observations suggest that while the R-CNN model shows some promise with its attention to relevant answers (as seen in its higher recall), it is prone to making many irrelevant predictions. Further model refinement is required to improve these metrics significantly. This highlights a potential area of weakness in the models comprehension of the visual and textual data and underlines the necessity for model refinement and further training.

The learning curves in figure 2 depict a consistent decline in loss for both training and validation datasets, indicating good generalization despite the small sample size. However, the limited training size (15,000 samples) due to computational constraints is a significant factor to consider, as it is significantly less than the total 82,000 available samples. This limitation is likely to have impacted the models ability to learn more complex features and generalize across a broader dataset.

The result of a small training size could heavily impact the models ability to generalize well, especially on classes that are poorly represented. Inspecting some cases, where the model predicts incorrectly provides interesting insights into the models predictive ability. Figure 3 shows 4 incorrectly predicted images.

Figure 3: Incorrectly predicted samples

For nationality prediction, the model may not have learned to associate visual features with complex human attributes like nationality, which typically require understanding beyond visual data. The prediction "cheese" suggests a potential misunderstanding of the question's context or a fallback to a common or random label when uncertain. In the object identification, predicting "umbrella" for "kite" might indicate that the model has learned similar features for objects that can appear in the sky. This confusion might arise from shape or color similarities. The cones color in the third sample being predicted as "white" instead of "orange" could be due to lighting conditions, or the model may not have learned to distinguish colors reliably in different contexts. Mistaking a "map" for "broccoli" in the fourth sample is a more significant error, possibly stemming from the models inability to capture fine details, especially if the image was resized, leading to a loss of resolution. It could also have picked up on "green" and associated that with "broccoli".

Reflecting on the influence of data pre-processing, the decision to resize images to 224x224 pixels could also have led to a loss of detail, which is critical for accurate predictions. For instance, distinguishing between a 'map' and 'broccoli' requires fine-grained visual cues that may have been lost during downsizing, causing the model to rely on broader features like color or texture, which can be misleading. The augmentation could also introduce noise that leads to incorrect predictions, such as misidentifying objects with altered appearance due to these transformations. The small training size and the possible loss of critical information during pre-processing suggest that the current performance is not a true reflection of their capabilities.

5.1 Sampling Bias Towards Distractors

During our investigation into the effects of our sampling strategy on the models learning process, a critical realization came to mind: the potential for an imbalanced subset that may favor distractor answers over ground-truth responses. When sampling a subset of this dataset, there is a risk that the selected questions might have a higher proportion of distractors compared to ground-truth answers. This could lead to training a model that is better at identifying distractors rather than the correct answers, thus skewing its learning and affecting its real-world performance. This has not been accounted for in the sampling, which could significantly affect its ability to discern subtly incorrect answers from correct ones. The dataset also has biases towards certain answer types, such as 'yes/no' responses and numerical answers, could skew the models predictive abilities, leading to over-representation of certain classes. However, it is likely that the sampling bias is the culprit for why the 'yes/no' accuracy is basically random guessing.

The models are susceptible to overfitting given the high dimensionality of the images and language data and the relatively small subset of sampled training data. Despite implementing regularization techniques, like dropout and weight decay, the models generalization to new data is limited. The learning rate scheduler and early stopping was designed to help mitigate overfitting, but the balance between model complexity and dataset size still remains a critical factor in achieving better generalization. The learning curves indicate a stabilization of training loss after a few epochs, especially in the R-CNN model. This could suggest that additional training data or epochs may have diminishing returns on performance improvement. It could also mean that the models have reached their capacity for learning from the data provided, making a review of model architecture or training strategies nessecary for further gains. But again, because of the sampled dataset size, conclusions on this are difficult to make.

5.2 Comparison with State-of-the-Art

Compared to state-of-the-art models, our models operate under significant computational constraints, reflected in their modest performance. These constraints limit the dataset size and training epochs, resulting in significant lower accuracy and F1 scores. Additionally, compared to state-of-the-art approaches, our model demonstrates the challenges of limited computational resources and dataset size. Agrawal et al. (2016) achieved a test accuracy of 44.24% using a larger dataset and a VGGNET19 and LSTM model, highlighting the impact of diverse datasets on performance. Yang et al. (2016) reported a higher accuracy of 57.6% with a stacked attention network, though the context of this result is unclear. Anderson et al. (2018) achieved an impressive 80.3% accuracy using Faster R-CNN with a complete dataset, while our models had accuracies of 29.53% and 30.38%. These comparisons illustrate the need for extensive datasets and computational power in developing competitive VQA models, suggesting potential improvements for our approach in future work.

6 Conclusion and Future Work

In conclusion, the faster R-CNN model showed balanced results across different categories, with the highest accuracy in the 'yes/no' category. The CNN model displayed slightly better overall performance with a marginally higher accuracy. The R-CNN model had higher recall but slightly reduced accuracy, indicating better identification of relevant answers but also a higher chance of false positives. However, due to the constraints of sampling the dataset with possible distractor bias, comparing CNN and Faster R-CNN models is not feasible. This distractor bias risks skewing the learning process and impairs a fair assessment of each models true potential. Therefore, future work should focus on utilizing the complete dataset for a thorough analysis, which would enable a fair comparison.

References

[1] Peter Anderson, 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. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. doi:10.1109/cvpr.2018.00636.

[2] Sota for visual question answering on vqa v2 (test-dev). https://paperswithcode.com/sota/visual-question-answering-on-vqa-v2-test-dev. Accessed: [date of access].

[3] Hangbo Bao, Wenhui Wang, Li Dong, Qiang Liu, Owais Khan Mohammed, Kriti Aggarwal, Subhojit Som, and Furu Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts, 2022.

[4] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, and Furu Wei. Image as a foreign language: Beit pretraining for all vision and vision-language tasks, 2022.

[5] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish Thapliyal, James Bradbury, Weicheng Kuo, Mojtaba Seyedhosseini, Chao Jia, Burcu Karagol Ayan, Carlos Riquelme, Andreas Steiner, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. Pali: A jointly-scaled multilingual language-image model, 2023.

[6] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. Vqa: Visual question answering. *International Journal of Computer Vision*, 2016. doi:10.1007/s11263-016-0966-6.

[7] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[9] PyTorch. Models and pre-trained weights. https://pytorch.org/vision/stable/models.html, 2023.

[10] VQA Team. Vqa challenge. https://eval.ai/web/challenges/challenge-page/514/leaderboard/1386, 2020.