Technical University of Crete
School of Electrical and Computer Engineering

Diploma Thesis

Visual Recognition of Text in Images for Question Answering using Deep Learning

Konstantinos Vlachos

Thesis Committee

Professor Michail G. Lagoudakis (School of ECE)
Professor Michalis Zervakis (School of ECE)
Professor Panagiotis Partsinevelos (School of MRE)

Chania, July 2024

Πολυτεχνείο Κρήτης
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Διπλωματική Εργασία

Οπτική Αναγνώριση Κειμένου σε Εικόνες για Ερωταπαντήσεις με χρήση Βαθιάς Μάθησης

Κωνσταντίνος Βλάχος

Εξεταστική Επιτροπή

Καθηγητής Μιχαήλ Γ. Λαγουδάκης (Σχολή ΗΜΜΥ)
Καθηγητής Μιχάλης Ζερβάκης (Σχολή ΗΜΜΥ)
Καθηγητής Παναγιώτης Παρτσινέβελος (Σχολή ΜΗΧΟΠ)

Χανιά, Ιούλιος 2024

Acknowledgements

I would like to express my deepest gratitude to Professors Michail G. Lagoudakis, Panagiotis Partsinevelos, and Michalis Zervakis for their invaluable guidance and support throughout this research project. Their insights and expertise have been instrumental in the development and completion of this thesis. I am profoundly grateful to my parents, Ioannis and Polixeni, for their support and encouragement. Their trust in me has been a constant source of motivation. Finally, I would like to thank my friends for their companionship and support, which have been crucial during this journey.

Abstract

Visual Question Answering (VQA) is a complex challenge that combines the domains of Computer Vision and Natural Language Processing. The key concept behind VQA is to be able to automatically answer questions, provided in the form of natural language text, about the content of a digital color image, provided also as part of the input. The answer is to be delivered also in the same form of natural language text. This diploma thesis explores the development of a VQA model, utilizing existing systems, trained on millions of data using deep machine learning techniques. More specifically, the two systems utilized are: EfficientNetB0 as the image feature extractor and BERT for question embedding. The feature maps generated by these two components are concatenated and are subsequently passed through a convolutional Neural Network architecture with two dense layers, which is responsible for making predictions. The goal of this model's architecture is to correctly classify inputs, consisting of a question and an image, to answers selected from a predefined set of 500 possible responses. Training the model involved leveraging Colab's Pro GPUs, experimenting with various configurations to optimize performance, and employing a range of callbacks for enhanced training stability. The resulting model demonstrated good performance in many cases, accurately recognizing objects, understanding scenes, and performing spatial reasoning to answer questions related to the input image. These results are illustrated through a series of correct and incorrect predicted answers on selected instances. Finally, limitations, future extensions and potential applications of the proposed approach are discussed.

Περίληψη

Η Απάντηση Ερωτήσεων μέσω Οπτικών Δεδομένων (Visual Question Answering, VQA) είναι μια σύνθετη πρόκληση που συνδυάζει τους τομείς της Υπολογιστικής Όρασης και της Επεξεργασίας Φυσικής Γλώσσας. Η βασική ιδέα πίσω από το VQA είναι να μπορεί να απαντά κανείς αυτόματα σε ερωτήσεις, που παρέχονται με τη μορφή κειμένου φυσικής γλώσσας, σχετικά με το περιεχόμενο μιας ψηφιακής έγχρωμης εικόνας, που παρέχεται επίσης ως μέρος της εισόδου. Η απάντηση πρέπει να παραδοθεί επίσης στην ίδια μορφή κειμένου φυσικής γλώσσας. Η παρούσα διπλωματική εργασία διερευνά την ανάπτυξη ενός μοντέλου VQA, αξιοποιώντας υπάρχοντα συστήματα, εκπαιδευμένα σε εκατομμύρια δεδομένα χρησιμοποιώντας τεχνικές βαθιάς μηχανικής μάθησης. Πιο συγκεκριμένα, τα δύο συστήματα που αξιοποιήθηκαν είναι: το EfficientNetB0 ως εργαλείο εξαγωγής χαρακτηριστικών εικόνας και το BERT για την ενσωμάτωση ερωτήσεων. Οι χάρτες χαρακτηριστικών που παράγονται από αυτά τα δύο στοιχεία συνενώνονται και στη συνέχεια τροφοδοτούνται σε μια συνελικτική αρχιτεκτονική νευρωνικού δικτύου με δύο πυκνά στρώματα, τα οποία είναι υπεύθυνα για την πραγματοποίηση προβλέψεων. Ο στόχος της αρχιτεκτονικής αυτού του μοντέλου είναι να ταξινομήσει σωστά τις εισόδους, που αποτελούνται από μια ερώτηση και μια εικόνα, σε απαντήσεις που επιλέγονται από ένα προκαθορισμένο σύνολο 500 πιθανών επιλογών. Η εκπαίδευση του μοντέλου περιελάμβανε κατάλληλη χρήση των Pro GPUs του Colab, καθώς και πειραματισμό με διάφορες διαμορφώσεις για τη βελτιστοποίηση της απόδοσης και την εφαρμογή μιας σειράς από callbacks για τη βελτίωση της σταθερότητας της εκπαίδευσης. Το μοντέλο που προέκυψε επέδειξε καλή απόδοση σε πολλές περιπτώσεις, αναγνωρίζοντας αντικείμενα με ακρίβεια, κατανοώντας σκηνές και εκτελώντας χωροταξικούς συλλογισμούς για να απαντάει σε ερωτήσεις σχετικές με την εικόνα εισόδου. Αυτά τα αποτελέσματα παρουσιάζονται μέσω μια σειράς σωστών και λανθασμένων προβλεπόμενων απαντήσεων σε επιλεγμένες περιπτώσεις. Τέλος, συζητούνται εκτενώς περιορισμοί, μελλοντικές επεκτάσεις και πιθανές εφαρμογές της προτεινόμενης προσέγγισης.

Contents

List of Figures

List of Abbreviations

AI . . . . . Artificial Intelligence
ANN . . . . . Artificial Neural Network
CNN . . . . . Convolutional Neural Network
DL . . . . . Deep Learning
GPU . . . . . Graphics Processing Unit
ML . . . . . Machine Learning
NLP . . . . . Natural Language Processing
RNN . . . . . Recurrent Neural Network
RGB . . . . . Red Green Blue
TPU . . . . . Tensor Processing Unit
VQA . . . . . Visual Question Answering
CV . . . . . Computer Vision
NN . . . . . Neural Network

1 Introduction

Contents

1.1 Introduction

Visual Question Answering (VQA) is an interdisciplinary field at the intersection of computer vision and natural language processing. It involves developing models that can understand and interpret visual content in images and generate accurate, contextually appropriate answers to questions posed in natural language. This task is highly challenging because it requires integrating two distinct types of data: visual data from images and textual data from questions. Moreover, the VQA models must not only recognize objects and scenes but also comprehend relationships, actions, and other abstract concepts depicted in the images. The motivation behind research in VQA is driven by the desire to create intelligent systems that can perceive and understand the world in a manner similar to humans. This capability has diverse applications, such as assisting visually impaired individuals

by providing audio descriptions of visual content, thereby enhancing accessibility. Additionally, integrating VQA systems into augmented and virtual reality environments can create enriched interactive experiences. Users can engage with their surroundings, asking questions and receiving context-aware information. The VQA challenge is a benchmark competition designed to evaluate the performance of VQA models. It provides a standardized dataset consisting of images, questions, and multiple human-annotated answers. The VQA dataset includes a diverse set of images and questions, encompassing a wide range of topics and difficulty levels. For instance, consider an image from the dataset featuring a tennis player, as shown in Figure 1.1. A VQA system could be tasked with answering questions such as:

- "What sport is the person playing?"
- "What piece of equipment is the person holding?"
- "What is displayed on the electronic board in the background?"
- "What is the color of the player's outfit?"
- "What might the number on the electronic board indicate?"
- "Is the player in motion or standing still?"

Figure 1.1: Image example from VQA train dataset [1].

1.2 Thesis Contribution

This thesis advances the field of Visual Question Answering (VQA) by developing a novel model architecture that integrates convolutional neural networks (CNNs) and transformer-based models. Our custom VQA model uses EfficientNetB0 for extracting rich visual features from images and a pre-trained BERT model for processing natural language questions. These features are concatenated and passed through a dense layer with ReLU activation, culminating in a SoftMax output layer that provides a probability distribution over 500 potential answers.

To ensure effective training and robust performance, we configured the model with categorical cross-entropy as the loss function and the Adam optimizer. We employed metrics such as categorical accuracy, top-K accuracy, precision, and recall to comprehensively evaluate the model's performance. Addressing class imbalance, class weights were assigned to emphasize minority classes. We also utilized ModelCheckpoint, EarlyStopping, and ReduceLROnPlateau callbacks to enhance training efficiency and prevent overfitting, while TensorBoard provided detailed monitoring and visualization of the training process.

Our experiments demonstrated the robustness and efficiency of our custom VQA model. We optimized the training process using high-performance GPUs and experimented with different batch sizes to balance performance and efficiency. The model showed a rapid increase in accuracy during the initial epochs, which later stabilized, indicating good generalization. Evaluation on the test dataset confirmed the model's capability to accurately interpret visual content and contextual nuances, effectively recognizing environments, objects, colors, and understanding relationships and interactions within scenes.

1.3 Thesis Structure

- Chapter 2 provides the foundational concepts of AI used in this thesis. This includes concepts of Computer Vision and NLP, such as Artificial Neural Networks, Convolutional Neural Networks, and Recurrent Neural Networks.

- Chapter 3 formally states the problem studied in this thesis and summarizes related work from the literature.
- Chapter 4 contains a detailed description of our Custom VQA model. We provide a detailed description of all the components it utilizes and its configuration. Moreover, we provide detailed insights into the VQA dataset and the required modifications that were needed to be done.
- Chapter 5 presents the experiments conducted to evaluate our VQA model's performance, highlighting its successful and unsuccessful cases.
- Chapter 6 summarizes the key findings of our research, reflecting on the effectiveness and limitations of our VQA model.

2
Theoretical Background

Contents

2.1 Artificial Intelligence – Deep Learning

One of the most rapidly emerging and promising fields of technology is Artificial Intelligence (AI). AI is one of Computer Science's major branches, which aims to create intelligent machines. These machines must be able to perform tasks that require some level of intelligence and are usually executed by humans. Depending on how intelligent these machines are, AI can be divided into three categories. In Artificial Narrow Intelligence (Weak AI) computers execute similarly or outperform humans in specific tasks, such as recommendation systems, strategic games, and self-driving cars. Furthermore, in Artificial General Intelligence (Strong AI) computers must be able to perform the same intellectual tasks as a human being. Last, in Artificial Super Intelligence computers are considered to be far more intelligent than every brightest human mind on Earth.

Machine Learning is a sub-field of AI which is devoted to making computers learn from data and be able to take decisions/actions by themselves. Its main focus is to take advantage of past data and try to understand patterns and connections between them in order to make new inferences from new data. This process is executed with little or no human intervention and demands great amounts of computations.

Machine learning can be split into three categories based on how algorithms learn. In Supervised Learning, both input and output are specified over a set of instances and the algorithm is fed with labeled training examples to generalize over all possible instances. In Unsupervised Learning, algorithms train on unlabeled data. Their goal is to scan through data inputs to find hidden patterns and correlations. Last, in Reinforcement Learning, algorithms decide on their own which steps to take and are given positive or negative feedback (rewards). So, the algorithm attempts to maximize rewards in the long term in order to achieve specific tasks.

Deep learning constitutes an essential subset of machine learning that has seen recent success in many fields, such as computer vision, speech recognition, and natural language processing. In traditional machine learning algorithms, essential features are extracted manually from raw input. Programmers have to decide the factors based on which essential features will be obtained from input, so the model could then make

predictions based on them. On the other hand, deep learning achieves automated feature extraction with no human intervention. Deep learning algorithms use several layers of data processing in order to extract features from raw input. Early layers are capable of identifying low-level features, such as edges on an image, whereas deeper layers are used to identify more complex features, such as entire shapes and objects.

The rise of big data and cloud computing has led deep learning to great success. Training deep learning models requires immense amounts of labeled data, as well as substantial processing power. This is due to the fact that deep learning algorithms continue to improve, as the dataset increases, as opposed to machine learning algorithms that reach a plateau in performance at a certain size of the dataset. Figure 2.1 shows a comparative overview of traditional machine learning and deep learning approaches in the context of image classification.

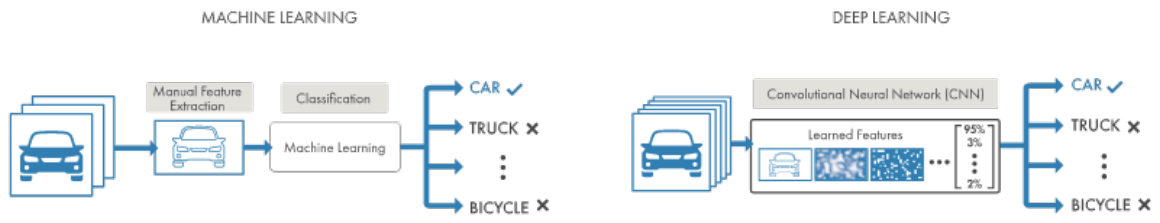


Figure 2.1: Differences between traditional machine learning and deep learning approaches to image classification [2].

2.2 Artificial Neural Network

2.2.1 Definition

Artificial Neural Networks are computing systems that are capable of mapping inputs to outputs by finding correlations between them. As the name suggests, these systems are inspired by biological neural networks and attempt to mimic how the human brain works. Typically, these networks consist of several layers and each layer has several nodes (neurons). First, there exist the input layer and the output layer, where the input is passed into the network and the output is delivered from it respectively. Then, between these layers, there exist one or more hidden layers, where calculations are made to extract features and patterns from the input data. Figure 2.2 provides a visual

representation of a basic neural network architecture, illustrating the movement of data from the input layer, through the hidden layers, to the output layer.

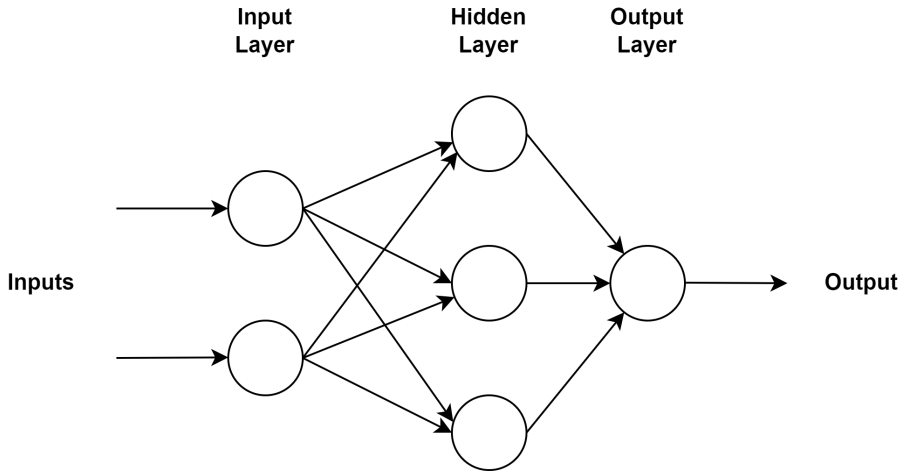


Figure 2.2: A simple neural network showcasing the flow of data from input layer to hidden layers to the output layer.

Each layer has its own number of neurons. The input data flow from the input layer to the output layer, with each neuron receiving inputs from the outputs of the neurons in the previous layer. Each connection between two neurons has a weight that is adjusted during training. The output of a neuron is calculated by taking the weighted sum of its inputs plus a bias term (another parameter adjusted during training). This value is then passed through an activation function to produce the neuron's output. This procedure is illustrated in Figure 2.3.

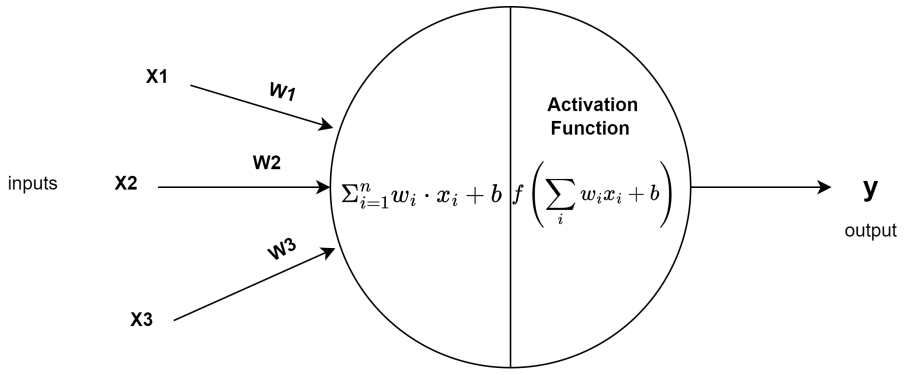


Figure 2.3: Illustration of a neuron in a neural network, where inputs x_i are multiplied by their corresponding weights w_i, summed, bias added, and passed through an activation function to produce an output y.

The purpose of the activation function (also called transfer function) is to introduce non-linearity in the network. Without the activation function, neural networks would only be able to approximate linear functions. But with non-linearity added, the model is able to generalize and find complex patterns between the input and the output. The most well-known and used activation functions are sigmoid, tanh, softmax, and ReLu, shown in Figure 2.4.

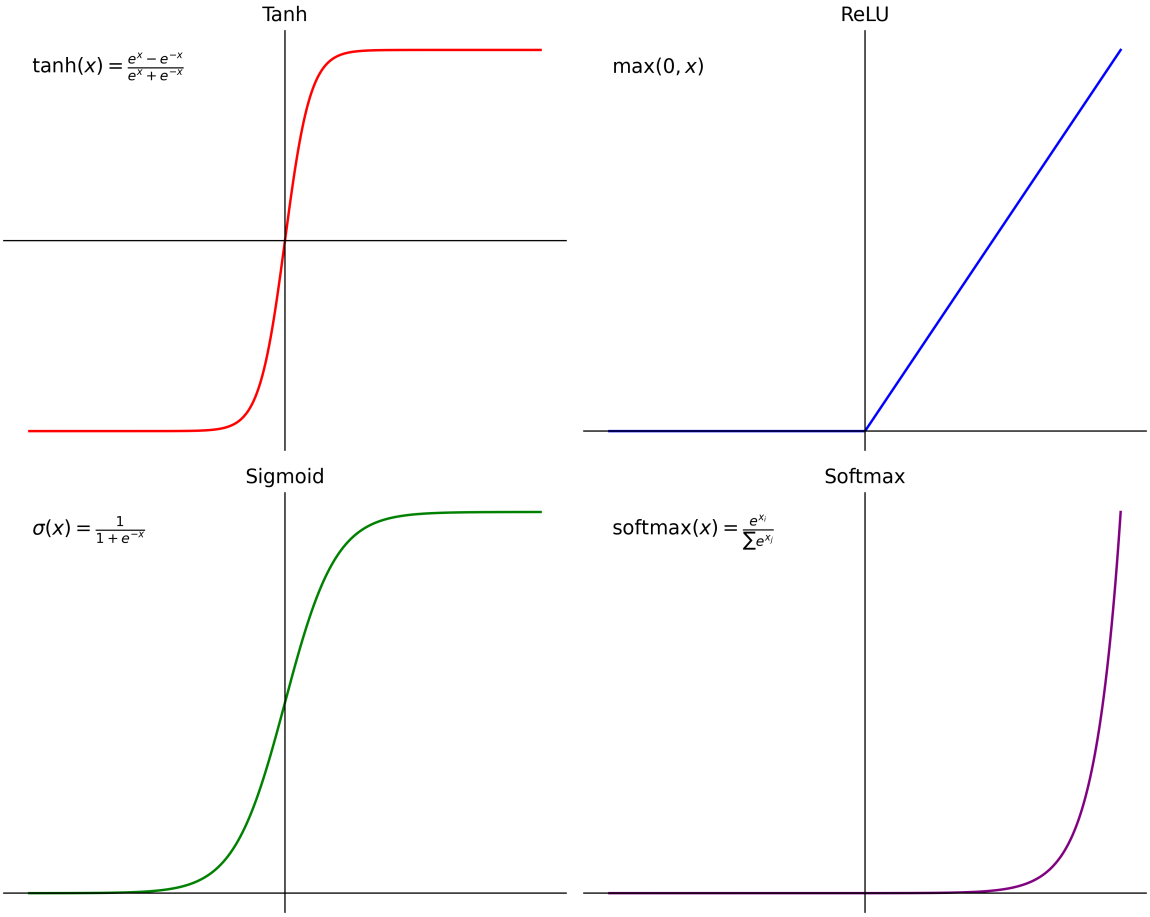


Figure 2.4: Activation functions commonly used in neural networks

2.2.2 Training

Neural Networks need to be trained on many data in order to learn to associate inputs with outputs and make good predictions. Training involves processing labeled examples one by one, whereby examples contain the input and the correct output (label). Learning is based on the trial and error method. Model makes a prediction on

the given input and this prediction is compared to the label. Based on the error of these two, the model adjusts its parameters with the goal of minimizing the error.

2.2.3 Forward Propagation

In detail, the model's parameters are initialized in most cases with random values. Then, the input data of the first example is inserted into the input layer of the model, where no calculations are executed in this layer. Next, data is passed into the first hidden layer, where each neuron computes its weighted sum of inputs and then its total output after applying the activation function. This process continues on the rest of the hidden layers and the output layer, where the final model's prediction is produced. The whole process of data flowing from input to output layer is called forward propagation.

2.2.4 Back Propagation

After completing forward propagation, the model compares the prediction with the label and measures their difference using a loss function. The objective of the training is to minimize the loss function. Once the loss is calculated, backpropagation (backprop) begins. Backpropagation is the process of propagating the error backward through the network, from the output layer to the input layer, in order to update the model's parameters (weights and biases).

Steps:

1. First, the gradient of the loss is computed with respect to the output of the neurons in the output layer. This gradient indicates the sensitivity of the loss function to a slight change in the output.

2. Then, by using the calculated gradient, we derive the gradient of the loss with respect to its inputs and weights. The weight's gradient indicates the adjustment needed to minimize the loss. At the same time, the input's gradient is propagated back to the previous layer to continue the process.

3. After calculating the gradients, the weights and biases are updated using an optimization algorithm. The most common optimization algorithm is Stochastic Gradient Descent (SGD), which updates the parameters by moving them toward the negative gradient. The weights "w" and biases "b" are updated based on these equations:

$$w = w - a\frac{\partial L}{\partial w}$$
$$b = b - a\frac{\partial L}{\partial b}$$

where "a" is the learning rate.

2.2.5 Convergence

The training process continues until the model's performance stabilizes, a state known as convergence. This means that either the loss no longer decreases significantly or a predetermined number of epochs has been reached (1 epoch = 1 pass over the training dataset). At this point, the model is considered to have learned the underlying patterns in the data and can make accurate predictions.

2.3 Computer Vision

2.3.1 Image Representation

A digital image can be represented as a two-dimensional array that contains color information. It consists of many small square elements called pixels. Each pixel has a numerical value that represents a specific color. This value is stored in a predefined number of bits called pixel depth. There are several types of images that are distinguished by pixel depth.

Pixels in Binary Images (also called black and white) can have two possible values, either "0" or "1". "0" represents the color black, and "1" the color white.

Pixels in Grayscale Images can have 256 different values as the pixel depth is 8 bits. These values represent different shades of gray color. Indicatively, "0" represents black, "128" represents gray, and "256" represents white.

In colored images, three separate 8-bit values are combined to form the color of one pixel. This is due to the RGB color space that forms any color by combining the three basic colors: Red, Green, and Blue. As a result, a colored image is represented by a 3-dimensional array instead of a 2-dimensional one, with the extra dimension storing the three RGB values for each pixel (see Figure 2.5).

Figure 2.5: Different types of visual data: binary image, grayscale image, and RGB image.

2.3.2 Advanced Topics in Computer Vision

Computer Vision (CV) is an essential field of Artificial Intelligence that has grown rapidly in recent years due to the rise of neural networks and the big data era. CV aims to make computers able to derive meaningful information and a high-level understanding of digital images and videos. It tries to mimic parts of the human visual system, so that it handles well common visual tasks.

The most popular computer vision tasks are classification, object detection, and instance segmentation (Figure 2.6).

Classification: Classify an image to a certain class.

Object Detection: Find what objects there are in the picture and locate them with bounding boxes.

Instance Segmentation: A more precise object detection where objects are distinguished pixel by pixel.

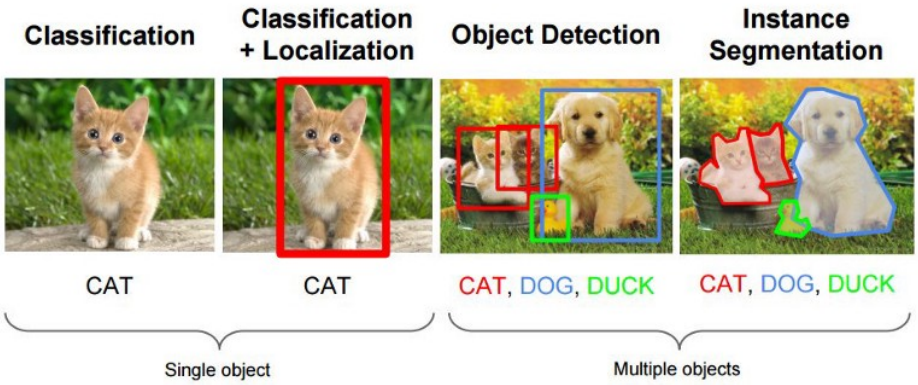


Figure 2.6: Computer Vision tasks [3].

2.3.3 Convolutional Neural Network

Convolutional Neural Network is a NN architecture used in Deep Learning which processes structured arrays of data (images in most cases). Its role is to handle very large inputs by applying filters in order to reduce size, without losing critical features. This makes it easier for the network to process the data and find patterns. CNNs excel at Computer Vision tasks with various state-of-the-art model architectures being inspired by CNN. In most CNN architectures there are typically three types of layers.

Convolutional Layer

A Convolution Layer is where the convolution operations (Figure 2.7) occur between the input data and the filter (Kernel). The outcome of the operation is a feature map that represents the features of the input. The filter is initially placed on the top left corner of the input's matrix, executing multiplication between overlapping values. The first element of the feature map arises from summing up these values. Moving on, the filter shifts right by a number of pixels, which is specified by a hyperparameter called Stride, calculating the next element of the feature map. After reaching the matrix's total width, the filter makes a similar shift (same Stride) downwards and starts again from the far left. This process continues until the whole input matrix is traversed [4].

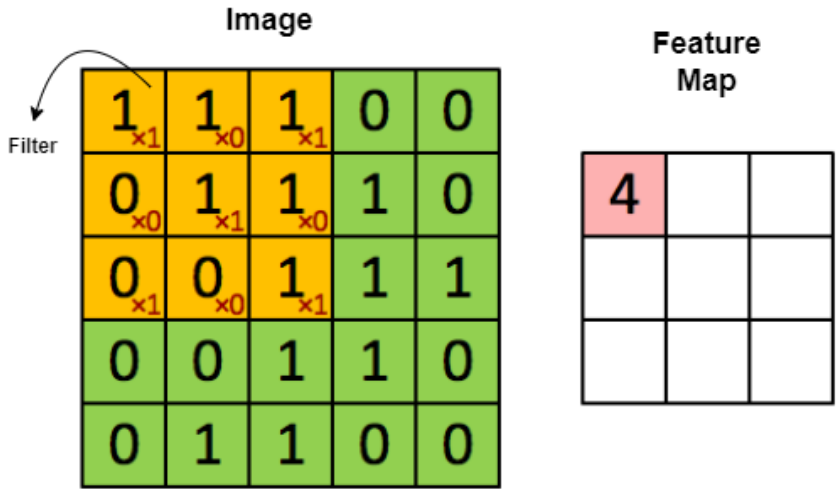


Figure 2.7: Convolution process in neural networks, where a 5x5 image is convolved with a 3x3 filter (stride = 1) to produce a 3x3 feature map [5].

The filter should always have the same depth as that of the input. So in the case of an RGB image (which contains three channels), the filter should have three dimensions (height, width, channels), where channels = 3, one for each color (Red - Green - Blue). The convolution operation is executed in every channel separately, producing three feature maps, and then they are summed up element-wise. The final result is one 2-dimensional (height, width) feature map that depicts the filter's reaction to the input image.

Typically, in a CNN, a large number of filters is applied to the input image. The process of applying multiple filters on the input image results in multiple output feature maps, each highlighting different patterns or features in the input image. These output feature maps can be further processed by additional layers in the neural network to extract higher-level features and eventually make predictions.

Pooling Layer

In a Pooling Layer, similar to a Convolution Layer, the goal is to reduce the spatial size of the feature maps produced by the Convolution. This reduction aims to minimize the computational resources needed for data processing, while maintaining the core information of the input. Moreover, the Pooling layer proves valuable in

extracting predominant features that exhibit rotational and positional invariance, thereby preserving the efficacy of the model's training process.

There are two types of pooling commonly used in CNN models: Max Pooling and Average Pooling. The primary goal of Max Pooling is to extract the most important features from the convolved feature maps. In this process, the largest value is chosen within a specific pooling zone, typically represented by a square or rectangular window (the filter), while the remaining values are discarded. Max pooling effectively reduces the spatial dimensions of the feature maps and enhances robustness to variations in position and orientation.

On the other hand, Average Pooling calculates the average value within each pooling region, providing a smoothed representation of the feature maps by averaging the values. Although average pooling also reduces the spatial dimensions, it may result in a loss of specific information compared to max pooling. These pooling techniques are illustrated in Figure 2.8, which shows how max pooling and average pooling are used in CNNs to downsample feature maps while retaining crucial information [4].

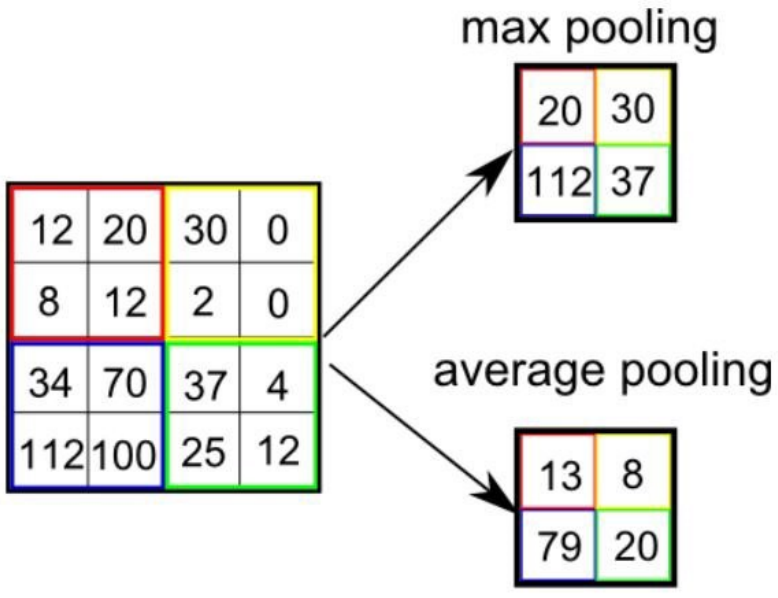


Figure 2.8: Max pooling and Average Pooling, techniques used in CNNs to downsample feature maps, while retaining important information [6].

Fully Connected Layer

A fully connected layer is typically placed in the final stages of a model's architecture.

In the domain of computer vision, a flattened feature vector is frequently used as the input to a fully connected layer. This vector is generated by reshaping the multi-dimensional feature maps produced by the preceding convolutional and pooling layers. As the name suggests, every neuron in the preceding layer is connected to every neuron in the fully connected layer, resulting in a large number of parameters that must be trained due to the dense connections. This layer is primarily responsible for capturing high-level semantic information and making predictions based on the learned features, as well as learning non-linear feature combinations from the produced feature maps. The structure of a typical Convolutional Neural Network (CNN) incorporating these layers is illustrated in Figure 2.9, which shows the flow of data from input through to the output layer [7].

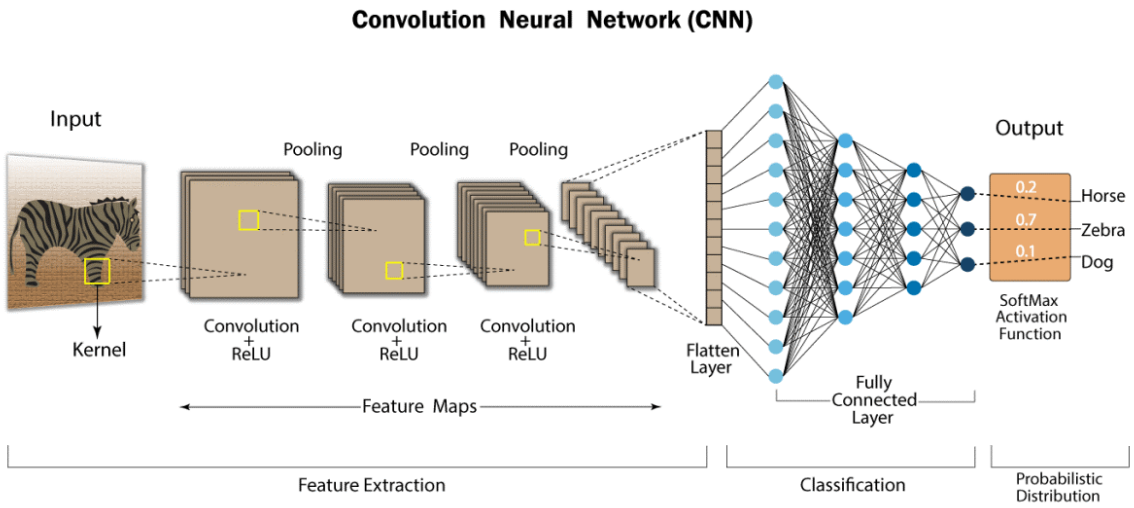


Figure 2.9: Structure of a Convolutional Neural Network for image classification [7].

2.4 Natural Language Processing

2.4.1 Definition and Scope

A significant subfield of Artificial Intelligence is Natural Language Processing (NLP), which focuses on the communication between human language and computers. This process entails creating algorithms and models that allow machines to comprehend, interpret, and produce human language with significance. At its core, NLP involves

designing and implementing computational techniques to process and analyze natural language data. It includes a wide range of tasks, including, but not limited to:

- **Language understanding**: Enabling machines to comprehend and extract meaning from written or spoken language.
- **Language generation**: Allowing machines to produce human-like text or speech based on given input or context.
- **Language translation**: Facilitating the translation of text or speech between different languages.
- **Language summarization**: Condensing and extracting the key information from a larger text.
- **Language sentiment analysis**: Identifying and classifying emotions, opinions, or sentiments expressed in text.

Based on these core tasks, NLP has found extensive applications in various fields.

- **Chatbots and Virtual Assistants**: NLP powers the conversational abilities of chatbots and virtual assistants, which can understand and respond to user queries accurately. This enables natural and human-like interactions with users for customer support, and information retrieval.
- **Speech Recognition**: NLP is the backbone of speech recognition systems, empowering voice-controlled devices and transcription services by converting spoken language into written text with remarkable accuracy.
- **Speech Synthesis**: NLP facilitates text-to-speech synthesis, that generates natural-sounding speech for applications like audiobooks, voice-overs, and accessibility tools.
- **Machine Translation**: NLP is utilized in machine translation systems, enabling the automatic translation of text or speech between different languages, and making global communication more accessible.

- **Sentiment Analysis**: NLP is being used to classify emotional tones in text, from positive to negative. This technology has the potential to provide crucial insights into understanding public opinions on social issues and also gain business insights into customer sentiment.

2.4.2 Key Concepts - Preprocessing

Text Preprocessing

Text preprocessing is a crucial step in NLP that involves cleaning and transforming raw text data into a format suitable for analysis and modeling. There are three primary objectives of this process. First, is to eliminate 'noise' or irrelevant information that could reduce the effectiveness of NLP models. Second goal is decreasing data's dimensionality by removing unnecessary features, thereby improving the efficiency of model training. Finally, text preprocessing tries to enhance data quality by correcting common issues, such as spelling errors and grammatical inaccuracies. By focusing on the most relevant information in the text data, this process ensures greater precision in subsequent analyses. Through effective text preprocessing, a solid foundation is established for more accurate and insightful NLP outcomes.

Preprocessing Techniques

Text preprocessing involves various techniques tailored to the specific needs of the task. Below, we explore the most commonly-used methods, each with a relevant example to demonstrate its application.

Tokenization

Tokenization is the process of splitting text into individual words or smaller sub-units, known as tokens, facilitating structured data for easier analysis. For instance, the sentence "The boy is a basketball player." would be segmented into tokens such as ["The", "boy", "is", "a", "basketball", "player", "."].

Case Normalization

Here, all characters in the text are converted to a uniform case (usually lowercase) to maintain consistency across the dataset. Thus, "Athens is the capital of Greece" would be normalized to "athens is the capital of greece".

Handling Punctuation

This process involves removing or replacing punctuation marks to simplify the data. For example, removing punctuation from "The answer is correct!!" results in "The answer is correct".

Stop Words Removal

This step involves removing commonly used words, known as stop words, which do not contribute significant meaning to sentences. This allows for a focus on the important terms in the data. For example, in the sentence "The boy likes going to the cinema", eliminating stop words, such as "the" and "to", results in "boy likes going cinema".

Word Embeddings

This advanced technique represents words in a continuous vector space, capturing semantic and syntactic relationships between them. It is crucial in deep learning models, facilitating the learning process from contextual word usage. For example, using Word2Vec, words like "dog" and "cat" are mapped closely in vector space compared to "dog" and "phone", showcasing the semantic similarity between the former pair.

2.4.3 Recurrent Neural Networks in NLP

Recurrent Neural Network

Recurrent Neural Networks (RNNs) emerge as a specialized subset of artificial neural networks engineered to manage sequential data or time series data. In contrast to traditional feedforward neural networks, which maintain a uni-directional flow of information and treat input and output as independent entities, RNNs incorporate a bi-directional flow. This unique design allows outputs from specific nodes to influence the input to those same nodes in subsequent iterations. This inherent loop within

the RNN architecture enables the network to retain past information and utilize an internal state or 'memory'. Such capabilities make RNNs especially adept at NLP tasks, which require sequential data handling and can involve an unfixed number of inputs, unlike feedforward neural networks that handle a fixed number of parameters. The process of unfolding an RNN over time, which visualizes how the network processes sequences, is illustrated in Figure 2.10 [8].

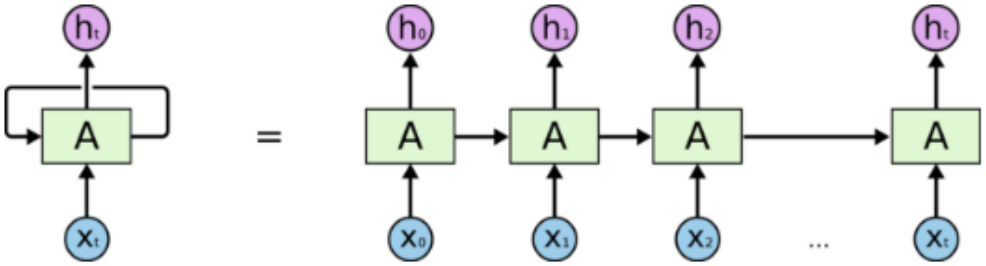


Figure 2.10: Process of unfolding a recurrent neural network (RNN) over time [9].

Forward Propagation

Forward propagation in an RNN is the process of computing outputs from a given sequence of inputs. For a given sequence, each input is processed one at a time, with the network producing an output and a hidden state. This hidden state, which encapsulates the 'memory' of the network, is then used along with the next input in the sequence to produce the subsequent output. This internal memory aids the network in understanding the context within a sequence.

For an input sequence x (where x_t is the input at time step t), the forward propagation can be characterized by the following steps:

Computing the Hidden State:

At every time step t, the new hidden state h_t is computed based on the previous hidden state h_{t-1} and the current input x_t.

$$h_t = \sigma(W_{xh} \cdot x_t + W_{hh} \cdot h_{t-1} + b_h)$$

Where:

- W_{xh} is the weight matrix for inputs.
- W_{hh} is the weight matrix for the previous hidden state.
- b_h is the bias for the hidden layer.
- σ is the activation function (commonly tanh or sigmoid).

Generating the Output:

The output y_t at time step t is then calculated based on the hidden state.

$$y_t = W_{hy} \cdot h_t + b_y$$

Where:

- W_{hy} is the weight matrix connecting the hidden state to the output.
- b_y is the bias for the output layer.

It is crucial to note that the same weight matrices (W_{xh}, W_{hh}, and W_{hy}) are reused across all time steps, a key difference from traditional feed-forward networks. This characteristic embodies the "recurrent" nature of the RNN and reduces the number of parameters, making them more suited for sequences of varying lengths.

Back Propagation Through Time

Traditional backpropagation excels in feedforward neural networks, where it navigates backwards through layers in a linear fashion, adjusting weights based on the error gradient with respect to the output. On the other hand, Back Propagation Through Time (BPTT) [10] extends the concept of backpropagation in order to be capable of dealing with the recurrent nature of RNNs, which process sequential data and where the output of a layer can loop back as an input. BPTT unfolds the RNN across time steps, treating the sequence of operations as a deep, albeit temporal, network. This unfolding allows for the sequential dependency of inputs and states to be considered.

First, the loss is computed at each time step based on the difference between the predicted output and the actual output.

$$L_t = \text{loss}(y_t, \hat{y}_t)$$

The backward pass also involves computing the gradients of the loss with respect to the model parameters by propagating errors backward through the unrolled network.

Gradient of Loss with respect to Output

$$\frac{\partial L_t}{\partial y_t} = \frac{\partial \text{loss}(y_t, \hat{y}_t)}{\partial y_t}$$

Gradient of Loss with respect to Hidden State

$$\frac{\partial L_t}{\partial h_t} = \frac{\partial L_t}{\partial y_t} \cdot W_{hy}$$

Gradient of Loss with respect to Parameters

- For W_{xh}:
$$\frac{\partial L}{\partial W_{xh}} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{xh}}$$
- For W_{hh}:
$$\frac{\partial L}{\partial W_{hh}} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{hh}}$$
- For W_{hy}:
$$\frac{\partial L}{\partial W_{hy}} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial y_t} \cdot h_t$$

Using the gradients computed in the backward pass, the weights are updated using gradient descent or any other optimization algorithm.

2.5 Tools and Frameworks

2.5.1 Tensorflow

Tensorflow is a free and open-source framework developed by Google for machine learning and primarily for deep learning applications. It is among the best machine learning frameworks with Pytorch, Scikit-learn and Caffe being other popular options. The foundation of Tensorflow's architecture are multidimensional arrays, called Tensors, representing every data within the framework. TensorFlow operations are carried out

within a computational graph, where tensors that flow between mathematical operations are represented by edges and nodes represent the mathematical operations. This structure enables efficient parallel computation across CPU and GPU architectures [11].

With tensorflow we can build, train and deploy various machine learning models with ease and run them on any device (desktop, mobile and cloud). Its integrated library offers a wide range of tools for different stages of the machine learning pipeline, from data preprocessing to model serving. It also provides abstraction and implementation of several utilities, such as loss functions, metrics and optimizers, that are used on neural networks. Finally, a large range of pre-trained models are available for use, which serves tasks like inference, transfer learning, and fine-tuning.

2.5.2 Keras

Keras is an open-source Deep Learning API written in Python and developed by Google. It is built on top of Tensorflow offering high-level building blocks for implementing neural networks as fast as possible. Its goal is to reduce cognitive load from developers and researchers in order to provide fast experimentation on neural network architectures. With keras we can build from simple artificial neural networks to complex and very deep convolutional neural networks (CNN) and recurrent neural networks (RNN). These network architectures are very popular in Computer Vision and NLP respectively [12].

2.5.3 Tensorboard

Tensorboard is a tool within the Tensorflow ecosystem designed for visualizing various aspects of machine learning and deep learning models' training and performance. This is achieved by tracking and visualizing metrics, such as loss and accuracy, over time. This allows users to monitor the training process, compare the performance of different models, and identify when a model has begun to overfit or underfit the training data. Also, TensorBoard provides a powerful way to visualize the computation graph of a model, displaying the detailed operations and structure of the neural network. This visualization aids in understanding the architecture's complexity and can help identify bottlenecks in the model design [13].

2.5.4 Colab

Colab (Colaboratory) is a hosted Jupyter notebook service owned by Google. Jupyter notebooks combine executable code along with enriched text (images, HTML, math formulas). Notebooks are separated into cells (blocks of code and text), where each cell can be executed individually on a web-based environment. It is used for Python code execution through the browser with no required setup. Colab is also integrated with popular machine learning frameworks, such as Tensorflow, PyTorch and Keras, making it ideal for AI projects. It provides access to powerful computing resources, such as GPUs and TPUs, that are offered for free and can be used for model training, enabling users to experiment with complex machine learning models and large datasets that require significant computational resources [14].

3
Problem Statement

Contents

3.1 Problem Definition

Visual Question Answering (VQA) is a challenging task that lies at the intersection of computer vision and natural language processing. It involves answering questions posed in natural language, based on the content of an image. The complexity of VQA arises from the need to understand and interpret both the visual content of the image and the semantics of the question and then generate an accurate and relevant answer. The idea for this thesis was inspired by the annual VQA Challenge. This challenge provides a relevant dataset, promoting the development of models that can understand and reason about visual and textual information effectively.

3.2 Related Work

The field of Visual Question Answering (VQA) has seen significant advancements, driven by the convergence of computer vision and natural language processing technologies. Early approaches to VQA primarily relied on fixed feature extraction methods combined with traditional machine learning classifiers. However, these methods struggled with the complexity and variability of real-world data. The introduction of deep learning has revolutionized the VQA landscape. Convolutional Neural Networks (CNNs) have become the standard for visual feature extraction, capturing detailed spatial hierarchies in images. These networks can provide robust representations of image content, which are crucial for accurate question answering [4, 15].

On the textual side, Recurrent Neural Networks (RNNs) were initially used to encode questions. These models could capture sequential dependencies in language, which are essential for understanding the context and nuance of questions. However, the advent of transformer-based architectures, like BERT, has marked a significant shift. Transformers have demonstrated superior performance in understanding complex language structures and are now widely used in VQA systems to process and encode questions [16, 8].

A notable trend in recent VQA research is the integration of attention mechanisms. Attention allows models to focus on specific parts of an image or question, improving the relevance and accuracy of the generated answers [17, 18].

4
Approach

Contents

Our approach to the VQA task is formulated as a supervised learning challenge using the VQA v2 dataset, which includes images paired with questions and multiple-choice answers. This involved creating a dataset focusing on the 500 most common answers to streamline the model's learning process and improve accuracy. We designed a custom neural network architecture that combines a pretrained EfficientNetB0 model as the visual feature extractor and a pre-trained BERT model for processing textual

questions. Both EfficientNetB0 and BERT are frozen during training to preserve their pre-learned capabilities and reduce computational costs. The extracted features are then fed into two dense layers, which are trained to classify the combined inputs into one of the 500 possible answers. To ensure the model's robustness, we implemented comprehensive data preprocessing, including image normalization and resizing, as well as the use of class weights to manage class imbalance.

4.1 Dataset

The dataset used for training and evaluating our custom neural network model is the widely-used VQA v2 dataset in the field of Computer Vision and Natural Language Processing. This dataset is built upon the MS COCO [19] images, which is a dataset rich in visual content. Its images are renowned for their diversity and for capturing everyday scenes with complex contextual information, featuring common objects in their natural environments. The goal is to expose our VQA model to a broad spectrum of visual scenarios in order to generalize across different contexts.

4.1.1 Questions

The questions in the VQA dataset are designed to test different levels of visual understanding and reasoning. There are multiple questions that are associated with every image in the dataset, approximately 9 questions per image. These questions were created by human annotators and can be divided into various categories.

- **Object Recognition**: Questions that are related to object identification and its characteristics within the visual frame (ex. "What color is the player's shirt?", "Does the guy have a tattoo?").

- **Action Recognition**: Questions that need to identify the action, the activity or the behavior taking place in the image ("What is the business man doing in the picture?", "How many people are surfing?").

- **Scene Understanding**: Questions that require synthesis of different visual elements to infer context and understanding of the entire scene ("Is this man a professional baseball player?", "Is there a shadow of a tree in the foreground?").

- **Spatial Reasoning**: Questions that involve spatial relationships between objects. ("Are the men on the sidewalk?", "What is on the pillow?").

4.1.2 Answers

The VQA dataset provides 10 answers for each question. Each answer is collected from different human annotators in order to achieve a solid foundation of the correct answer. The answer spectrum can have a wide variety. The given answers can be single words such as "yes", "no", "white, "five", or small phrases, like "playing baseball", or even entire sentences, such as "man in blue shirt". The VQA training set consists of 82,783 images, 443,757 questions and their corresponding answers. From the 10 answers provided for each question we kept as ground truth the most popular one. So, from those 443,757 answers we calculated a dictionary with all the unique answers as keys (23,390 total unique answers) and number of times appeared on the dataset as values. We kept the 500 most common answers and its relevant training examples (total: 367,806) in order to reduce dataset's imbalance and be more feasible for our model to learn to classify between the 500 answers. We can see in Figure 4.1 that with these 500 possible answers the most questions of the VQA training set can be answered correctly. By reducing the answer pool of our model, we strive to make it more accurate on the most frequent answers.

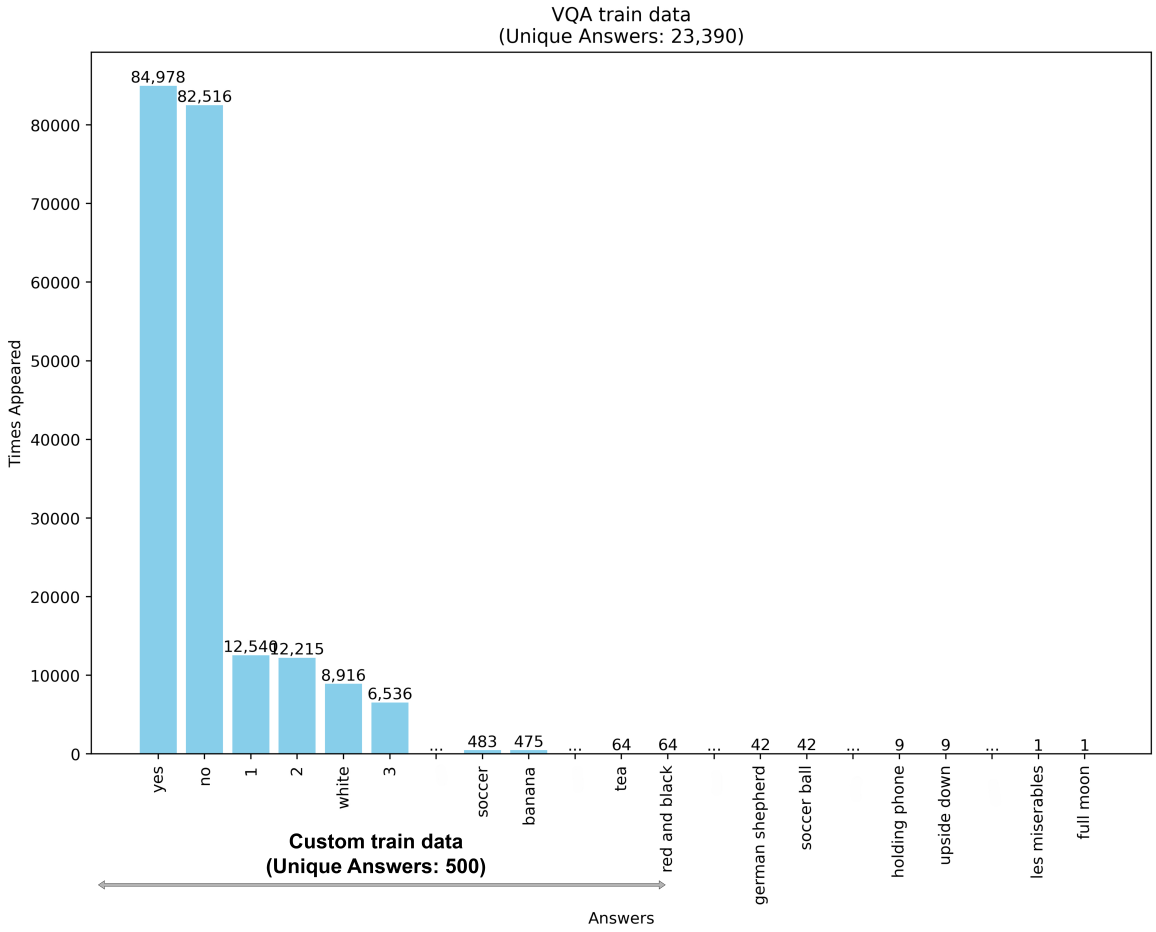


Figure 4.1: Frequency of the most common answers in the VQA training dataset. Our custom training dataset focuses on the top 500 most common answers.

The validation set consists of 40,504 images, 214,354 questions and their corresponding answers. From those 214,354 examples we kept only those that can be answered with the 500 most common answers of the training dataset (176,831 examples). Considering that our computation demands are already high due to the large training dataset we chose 30,000 examples for our Custom validation set (Figure 4.2). This was determined to be the sweet spot between computation efficiency and diversity on validation set (approximately 10% of train dataset size). On these 30,000 examples there are 6,877 unique images.

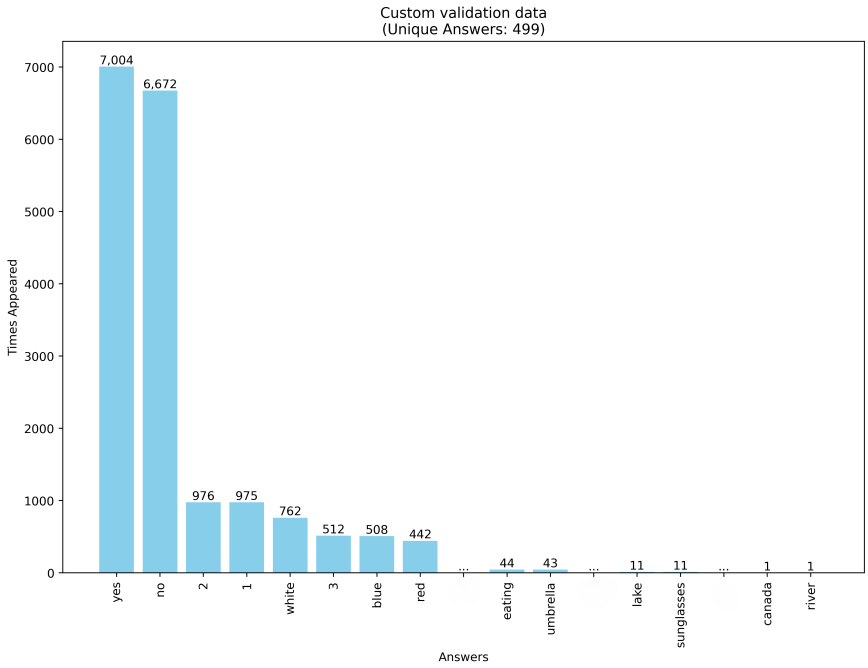


Figure 4.2: Frequency distribution of answers in the VQA validation dataset

The test set does not include the corresponding answers, because it was used at the VQA challenge in order to measure the effectiveness of the submitted models. So, we derived from the VQA validation set 30,000 examples as our validation set and another 30,000 examples for our test set (Figure 4.3).

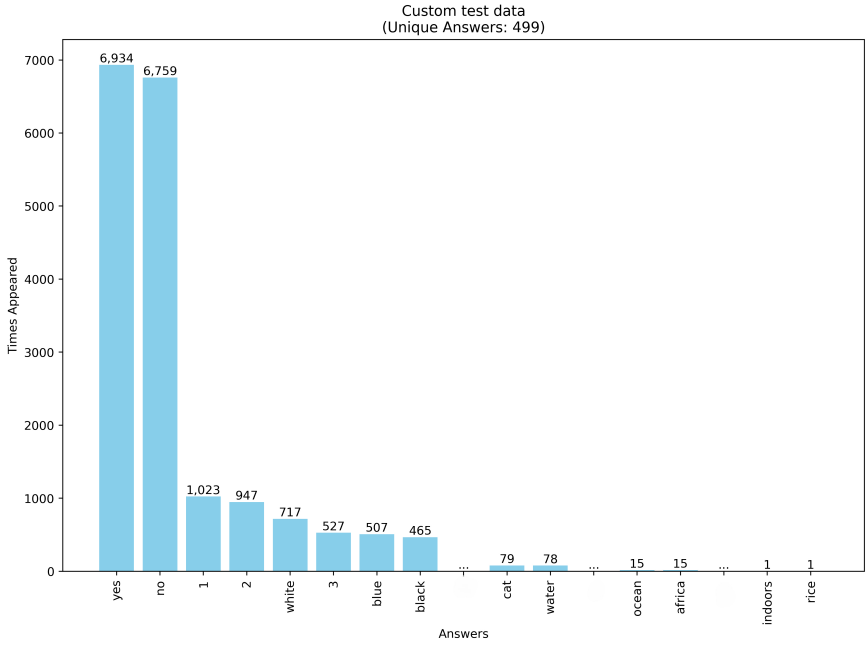


Figure 4.3: Frequency distribution of answers in the VQA test dataset

4.1.3 Data Preprocess

Training examples were loaded in batches, because the dataset was very large to fit in the RAM all at once. So for each batch, we first load the images by using the corresponding image_ids to create the image_paths. The images were then resized to specific dimensions (512×512 pixels) in order to ensure uniformity and compatibility with the input architecture of our model. Then the images were normalized to a range of [0,1] by dividing with 255, which is the maximum pixel value for an 8-bit image. This helped with enhancing the convergence rate during training by scaling the pixel values to a common range. Lastly, at the beginning of each epoch, the training examples were randomly shuffled to prevent overfitting to the order of the training data and help the model to generalize better.

The questions didn't need any special preprocessing, because our model accepts as input the whole question, as it is stored in the dataset. So, it wasn't required neither tokenization (splitting the question into words) nor encoding (transforming each token into a numerical format). This is because one of the main components of our custom model is a pretrained BERT model that handles whole questions as input.

For the preprocessing of the answers, several steps were implemented. First, from the 500 most common answers we created a dictionary (answer_to_index) that relates each answer to a unique index, starting from 0 to 499. This is mandatory, because neural networks are only capable of producing numerical values and not entire strings as predictions. Then, we also created the reversed dictionary (index_to_answer) in order to be able to transform the model's prediction, which is a number, to the corresponding answer. The last step is to transform answers_index to one-hot vectors. This is a common step in classification tasks, since it standardizes the data for efficient processing. One-hot encoding represents each answer as a binary vector, a vector of size 500 having all zero values, except for the corresponding index having value '1'. For example, answer_index = 4 will be transformed to answer_one_hot = ['0', '0', '0', '0', '1', ... , '0', '0'].

4.2 Image Feature Extraction

EfficientNet is a family of convolutional neural networks designed to achieve an optimal balance between accuracy and efficiency. The EfficientNetB0 model is the baseline of this family. Each of the subsequent models (EfficientNetB1, ... , EfficientNetB7) has increased depth, width, and resolution that are uniformly scaled. These models achieve slightly higher accuracy from the baseline, but with higher computation needs. For that reason and due to the very large dataset, we selected the EfficientNetB0 as our Image Feature Extractor. We used a pretrained version from Keras framework that it is trained on the ImageNet dataset [15].

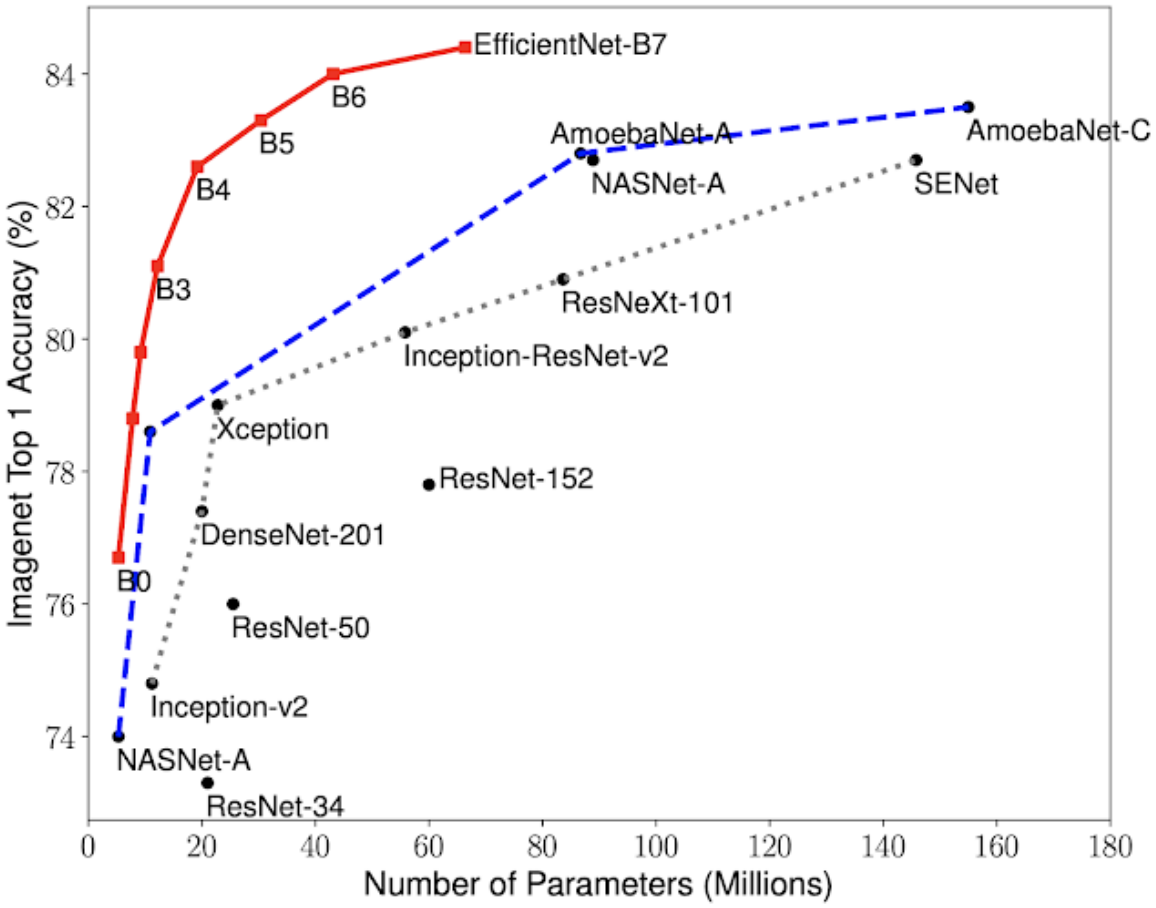


Figure 4.4: Performance comparison of EfficientNet models with other architectures based on ImageNet Top 1 Accuracy and parameter count[15]

4.2.1 Mobile Inverted Bottleneck Convolution (MBConv)

Mobile Inverted Bottleneck Convolution is a critical component in EfficientNetB0 that is designed to enhance efficiency and performance. It starts with an expansion phase that increases the dimensionality of the input through pointwise convolution. It then employs a depthwise separable convolution to process each channel independently and reduce the number of parameters. After that, the projection phase reduces the dimensionality back, to match the original input by applying another pointwise convolution. Finally, a residual connection, which is a direct connection from the original input to the output of the projection phase, helps preserve information and improve gradient flow. This means that the network can learn from the original input, as well as the transformed input, leading to better performance. With this structure, the number of parameters is effectively reduced, while concurrently the accuracy is improved.

4.2.2 EfficientNetB0 Architecture

As shown in Figure 4.5, the EfficientNetB0 network starts with an input image with shape (512, 512, 3), which represents the dimensions of the image: height, width, and color channels (RGB). The initial layer, a standard convolutional layer with a 3×3 kernel, is strategically designed to capture basic features like edges and textures, laying a solid foundation for the network.

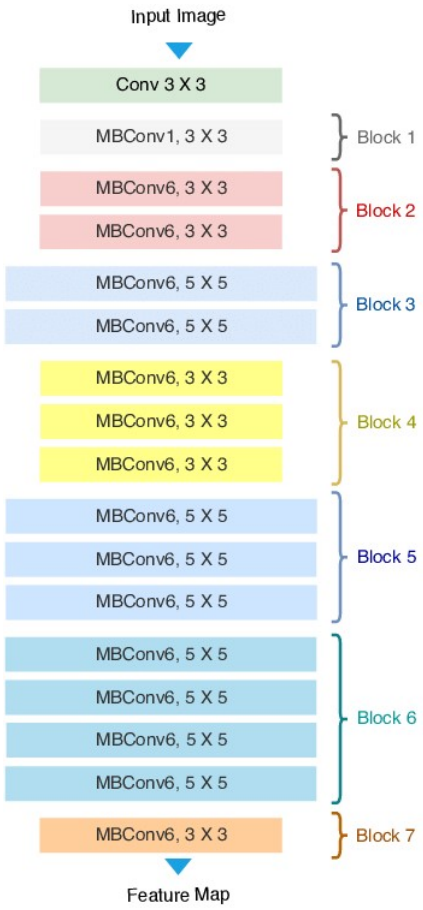


Figure 4.5: Architecture of EfficientNetB0 [20].

The network is divided into seven blocks, each containing one or more MBConv layers described below. As the network progresses, the depth and complexity of the layers increase.

MBConv1, 3×3: This is the first Mobile Inverted Bottleneck Convolution (MBConv) block with a 3×3 kernel and a depth multiplier of 1. It efficiently captures features while reducing the number of parameters.

MBConv6, 3×3 / 5×5: Subsequent MBConv blocks use a higher depth multiplier (6) and vary in kernel sizes (3×3 or 5×5). These blocks are designed to capture more complex features at different scales.

The output obtained after the final block is a feature map representing the acquired features from the input image, with shape (16, 16, 1280). This feature map is crucial for the VQA task as it encapsulates detailed visual information that will be combined with question embeddings derived from a pre-trained BERT model.

Together, these features enable the model to make informed predictions about the answer by understanding both the visual context provided by the image and the semantic context provided by the question. [20].

4.3 Word Embedding - BERT

The Transformer is a neural network architecture that leverages self-attention mechanisms to process data sequences, allowing for the parallelization of computations and capturing long-range dependencies within the data. Unlike traditional recurrent neural networks (RNNs) that process inputs sequentially, the Transformer processes the entire input sequence simultaneously, which significantly speeds up training and improves the ability to model complex dependencies. Key components of the Transformer include multi-head self-attention mechanisms, which allow the model to focus on different parts of the input sequence, and positional encodings, which provide the model with information about the position of each word in the sequence.

Attention is a mechanism that allows models to weigh the importance of different parts of the input sequence. It helps the model decide which parts of the input are more relevant for generating a specific output. In self-attention, each word in the input sequence is compared with every other word to compute attention scores, which indicate the importance of each word relative to others. These scores are used to create weighted representations of the input. [18].

Multiple attention heads operate in parallel, each learning to focus on different parts of the input sequence. Each attention head performs its own set of attention calculations and produces a different representation of the input. These representations are then concatenated and linearly transformed to form the final output [18].

In our Custom VQA model, we utilized a pretrained BERT model from TensorFlow Hub. Specifically, we used the base version of BERT, which has approximately 110 million parameters. This model is pretrained on a large corpus of text data in order to capture rich linguistic information. During training, the parameters were frozen to leverage the pretrained knowledge without updating the weights. In order to embed questions, there were needed the preprocessor and encoder modules [16].

Preprocessor: The preprocessor module is responsible for transforming raw text inputs into a suitable format for BERT. This includes tokenization (splitting text into word pieces) and generating segment and position embeddings.

Encoder: The encoder module is the core of the BERT model, which consists of 12 layers (L-12), each with 768 hidden units (H-768) and 12 attention heads (A-12). This architecture allows BERT to capture various linguistic features and dependencies across the input text. The encoder transforms the preprocessed tokens into dense vector representations. It produces word embeddings with shape (128, 768) where 128 stands for question length (smaller questions are padded) or one single embedding with shape (768) that represents the summary of the question.

4.4 Whole Model's Architecture

Visual Question Answering (VQA)

Visual Question Answering (VQA) is a complex task that requires a model to comprehend and integrate both visual and textual inputs to generate accurate answers to image-based questions. Our Custom VQA model employs convolutional neural networks (CNNs) for image feature extraction and transformers for processing textual data.

As the image feature extractor, we chose the pre-trained EfficientNetB0 model. This model was selected for its ability to extract rich visual features from input images with low computational cost and high accuracy. For handling the textual data, a pre-trained version of BERT was employed, known for its state-of-the-art performance in various NLP tasks. BERT's role in the VQA task is to capture the nuances of the questions being asked, enabling the model to better understand and respond to the queries [17, 21, 22]. The architecture of our Custom VQA model, combining these elements, is depicted in Figure 4.6.

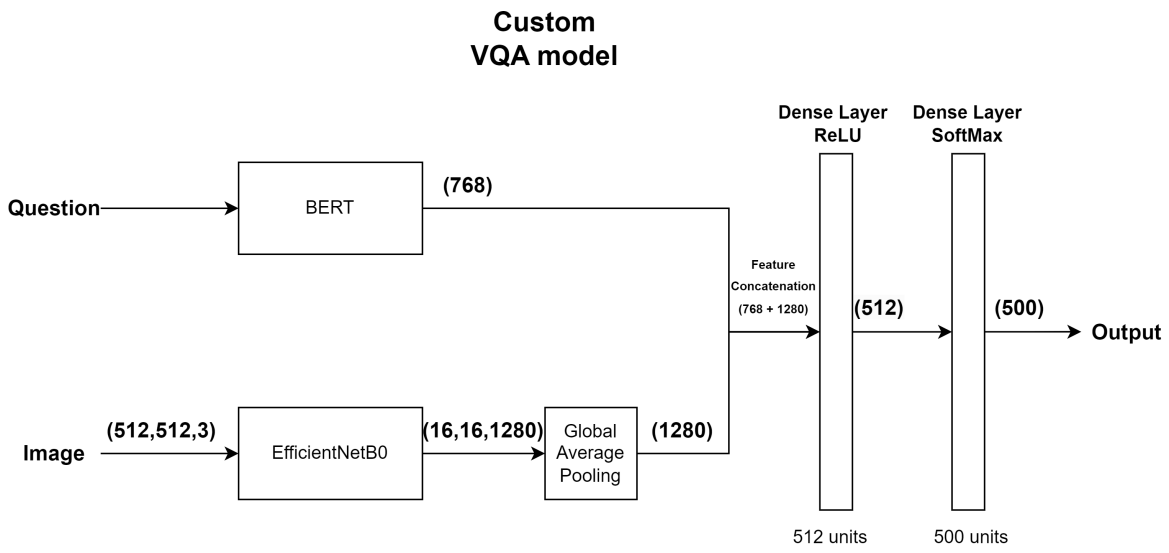


Figure 4.6: The architecture of our Custom VQA model.

Input Layers

- **Image Input**: The preprocessed image with shape (512, 512, 3)
- **Question Input**: Whole Question as String

Image Feature Extraction

- The input image is passed through the pretrained EfficientNetB0 model whose weights are frozen (not updated during training). The reason for freezing the weight of the model is because we want to keep the trainable parameters as low as we can and not waste computing power to fine tune parameters that already produce accurate image features. The resulting output is a feature map of shape (16, 16, 1280).
- Global Average Pooling is applied to the feature map to obtain a one-dimension vector that represents image features with shape (1280).

Text Feature Extraction

- The question input is processed through the BERT model and it generates embeddings of shape (768). These embeddings represent the meaning of the question, as feature maps represent the meaning of the image.

Feature Merging

- The pooled image features of shape (1280) and the question embeddings of shape (768) are concatenated to form a combined feature vector with shape (2048).

Fully Connected Layers

- The merged features are passed through a dense layer with 512 units (neurons) and ReLU activation to learn higher-level representations. This dense layer is crucial as it contains the majority of the model's trainable parameters, allowing the network to learn higher-level representations from the extracted features.
- The final output layer is a dense layer with SoftMax activation and 500 units, which corresponds to the number of possible answers the model can predict. These 500 answers were preselected based on their frequency in the training dataset, ensuring that the model is trained to predict the most common and relevant answers. This selection process was aimed at balancing the dataset and making the training process more efficient. In particular, this layer produces a probability distribution over the possible answers, with the answer having the highest probability being the model's prediction.

4.5 Model Compilation

The compilation of the VQA model is a critical step that configures the model for training by defining the loss function, the optimizer, and the evaluation metrics.

4.5.1 Loss Function

In multi-class classification tasks, in our case VQA, in most cases the categorical_crossentropy is applied as loss function. On forward-pass, each training example belongs to a corresponding category (answer) and by flowing through the model a prediction of probabilities for each answer is generated. The categorical_crossentropy calculates the difference between the true and predicted answer distribution and it effectively penalizes the model, when the predicted probability for the correct answer is low.

4.5.2 Optimizer

The optimizer is used to minimize the loss function during training. Adam [23] constitutes an adaptive learning rate optimization algorithm designed to handle sparse gradients. It computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. It is suitable for training deep neural networks and it is widely preferred for its efficiency.

4.5.3 Evaluation Metrics

To measure the performance of our VQA model during training and validation, we use a combination of the metrics described below.

Categorical Accuracy

The Categorical Accuracy metric computes the frequency of model's predictions that match the true labels. It is a straightforward metric that indicates how often the model's predicted answer is correct.

Top - K Categorical Accuracy

The TopKCategoricalAccuracy metric, with $k = 5$, measures the accuracy of the model by checking if the true answer is among the top 5 predicted answers.

Precision

The precision metric computes the ratio of true positive predictions to the total number of positive predictions made by the model. This metric is especially useful when working with unbalanced datasets.

Recall

The Recall metric computes the ratio of true positive predictions to the total number of actual positive instances in the dataset. Recall is crucial in understanding how well the model captures all relevant positive instances.

4.5.4 Class Weights

Class imbalance is a common issue in training neural networks, particularly in multiclass classification tasks, where some classes have significantly more samples than the others. This imbalance can cause the model to be biased towards the majority classes, in our case towards “yes” and “no”, which are the classes with far more samples. This leads to poor performance on minority classes, like some relatively rare answers, such as “tea” and “red and black”. To address this issue, class weights can be applied during training.

Class weights introduce an option to assign different emphasis to each class during training. By assigning higher weights to the minority classes, the model is higher penalized for misclassifying them. To apply this, the loss function needs to be modified by class weights. In the case of categorical cross-entropy loss, the class weight is multiplied by the loss contribution of every sample. Due to the increased cost of errors in minority classes, the model is encouraged to focus on these classes in order to increase its accuracy. Class weights can be calculated for each class using the inverse of the class frequency:

$$\text{Class\_Weight}_i = \frac{\text{total examples}}{\text{examples of Class } i}$$

4.5.5 Callbacks

During the training phase of our VQA model, different callbacks were used to improve the training process and achieve the best performance. Callbacks include features, such as preserving the optimal model weights, tracking training progress, and modifying learning rates. In this section, we provide a comprehensive explanation of the primary callbacks employed in our model training.

Model Checkpoint

The ModelCheckpoint callback is utilized to save the model's weights at different intervals throughout the training process. This ensures that we can preserve the best-performing model according to a particular metric, specifically the validation loss (val_loss). Monitoring val_loss is essential for ensuring that the model generalizes well to unseen data. The model weights are saved, only when there is a decrease in validation loss, thus preventing overfitting and guaranteeing that the best model is retained.

Early Stopping

The EarlyStopping callback is used to terminate the training procedure, if the model's performance fails to demonstrate improvement over a specified number of epochs, as determined by the patience parameter. This aids in conserving computational resources and mitigating the risk of overfitting. Training in our configuration will cease, if there is no enhancement in validation loss for two consecutive epochs. The patience was limited to only 2 epochs due to the dataset's substantial size, which demands significant processing resources.

Learning Rate Reduction

The ReduceLROnPlateau callback is used to decrease the learning rate, when the validation loss has stopped improving. This can boost the model's convergence towards

a better local minimum. Specifically, the learning rate is decreased by a factor of 0.2, if there is no improvement in validation loss for 2 consecutive epochs.

TensorBoard

TensorBoard is a common tool for visualizing the training process, monitoring metrics, and identifying errors in the model's architecture or in hyperparameter selection. The TensorBoard callback was configured to log training metrics after every batch and to generate histograms every 9,000 batches. This configuration produced several histograms on each epoch that helped us inspect the model's performance.

5 Experiments and Results

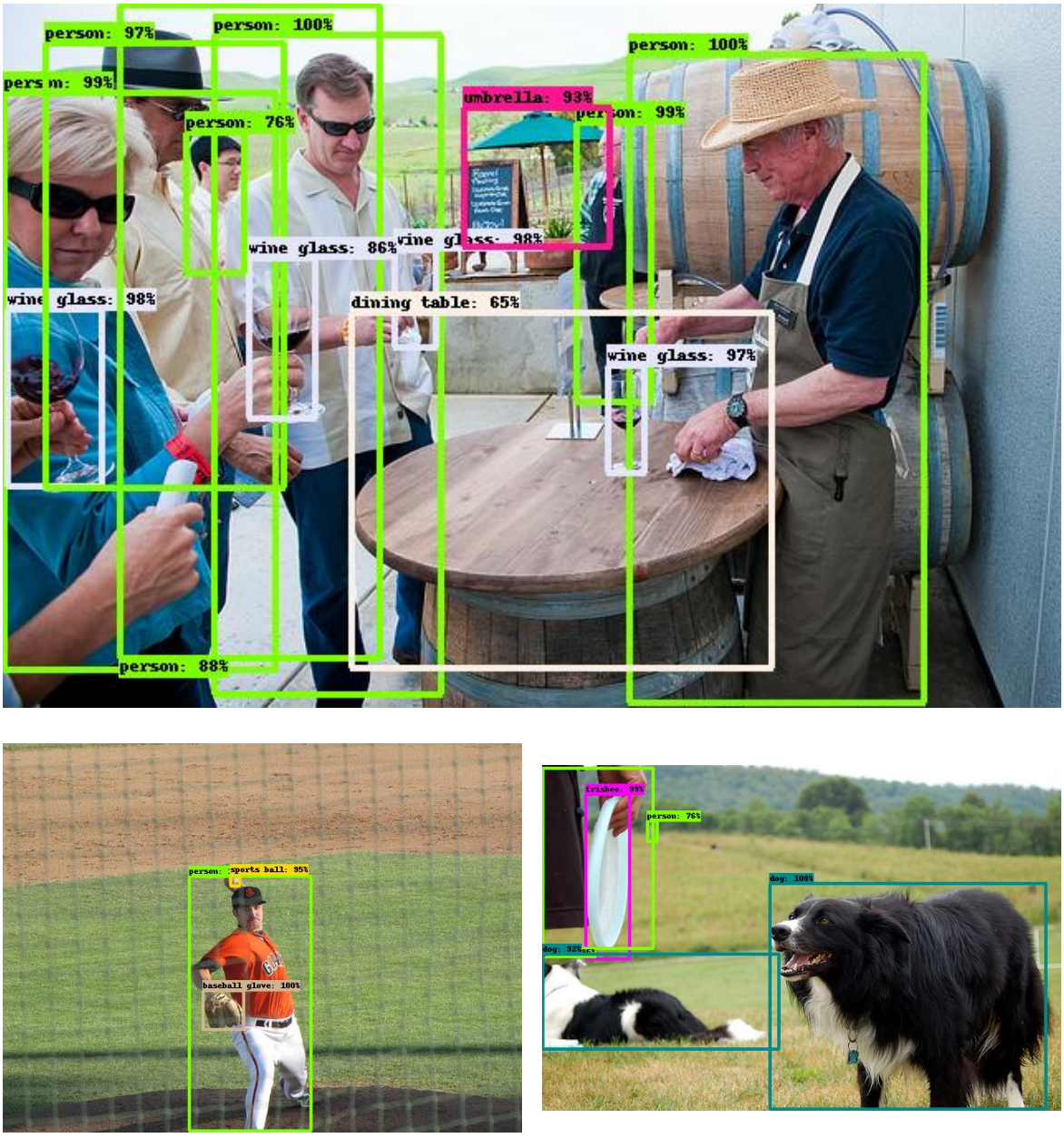
Contents

5.1 Object Detection

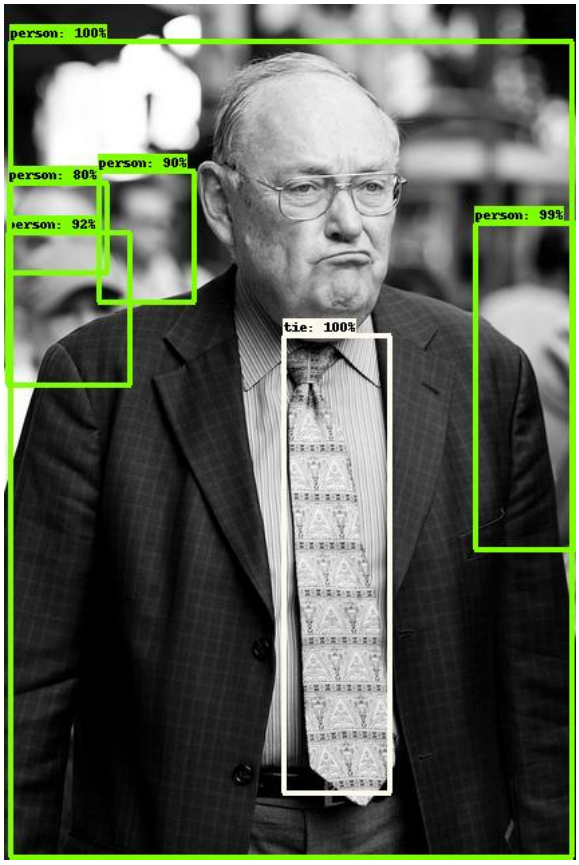
First, we wanted to validate EfficientNetB0's effectiveness as an image feature extractor in the VQA model and so we conducted object detection experiments using EfficientDetD4 on a set of VQA test data images. The goal was to demonstrate that the features extracted by EfficientNetB0 are robust and informative enough to support accurate object detection and therefore to produce accurate image representations for our VQA model.

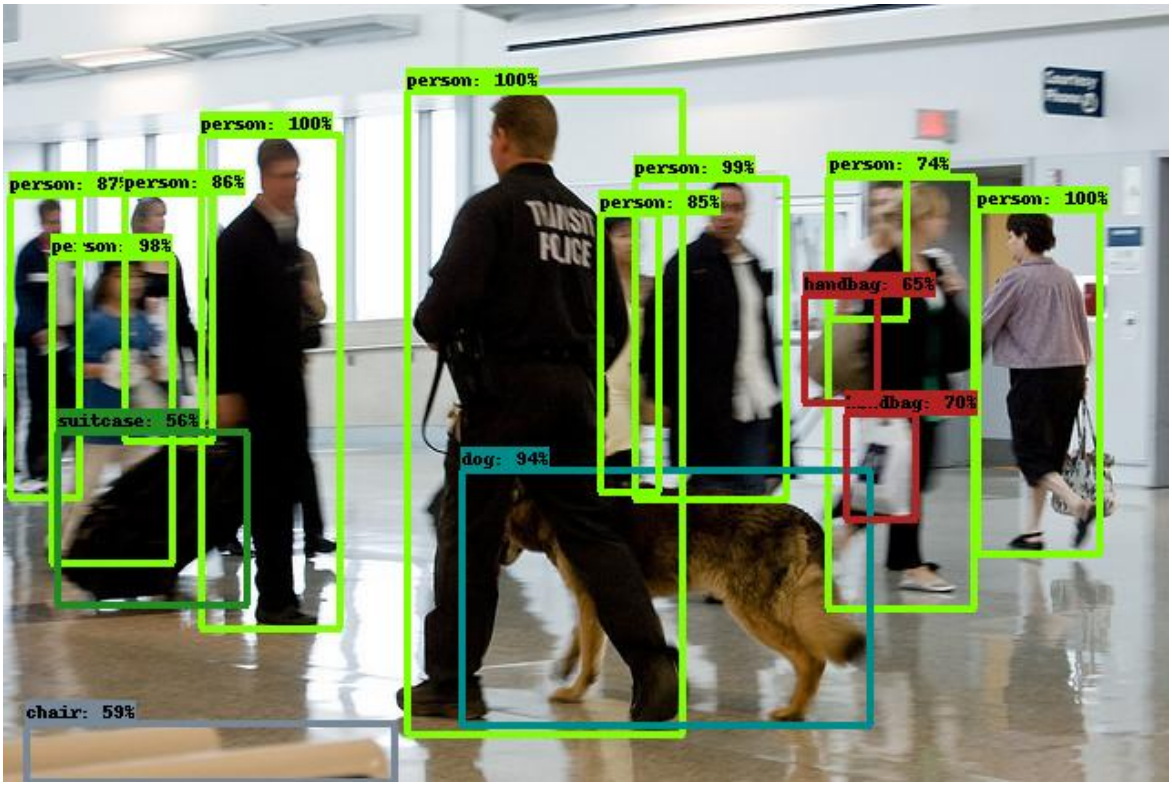
EfficientDetD4 is part of the EfficientDet family of object detectors. It incorporates EfficientNet as its backbone, benefiting from the efficient feature extraction capabilities of EfficientNet. It basically leverages EfficientNet's produced feature map to accurately detect and classify objects.

We utilized a pretrained version of EfficientDetD4 implemented in Tensorflow Object Detection API. This model was trained on the COCO 17 dataset and can detect 80 different object categories. In our experiment, we displayed object predictions with a probability greater than 50%. Objects detected with this confidence level were highlighted and distinguished using different colors in Figure 5.1.




person: 100%
backpack: 96%
skis: 94%
skis: 97%


person: 100%
person: 90%
person: 80%
person: 92%
person: 99%
tie: 100%


person: 100%
person: 100%
person: 87%
person: 86%
person: 98%
suitcase: 56%
person: 99%
person: 85%
person: 74%
person: 100%
handbag: 65%
dog: 94%
chair: 59%

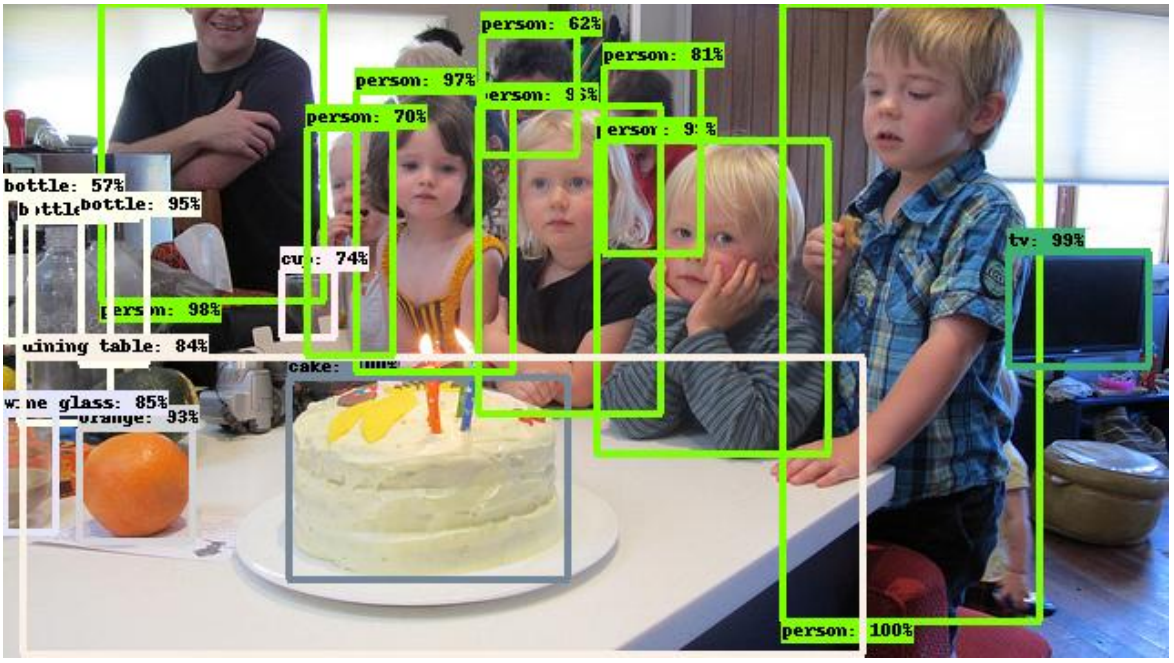


Figure 5.1: EfficientDetD4 accurately identifies and classifies objects in images from VQA test dataset, marking each detected object with distinct colors and labels based on the probability of detection.

5.2 Training Configuration

Upon evaluating the results of object detection, we observed impressive performance across various test images. The model effectively detected and classified most of the appeared objects. These promising results underscore the capability of EfficientNetB0 to extract meaningful features from images.

For training our custom VQA model, we had to utilize **Colab's** Pro GPUs to ensure reasonable training times due to the complexity of the VQA task and the large dataset. So, we experimented and compared the standard CPU and various GPU options, including T4 and the more advanced L4 GPU, which offers excellent computational power. The derived results for each epoch's training time were:

$$\begin{aligned} \text{CPU} &\rightarrow 280\text{h} \\ \text{T4} &\rightarrow 8\text{h} \\ \text{L4} &\rightarrow 1\text{h}30' \end{aligned}$$

We also tested different batch sizes, ranging from small values (4, 8) to larger values (16, 32, 64), because they affect training time and performance. Generally,

larger batch sizes mean faster training time, but require more GPU memory. After testing various configurations, we determined that using the L4 GPU with a batch size of 8 provided the best balance of performance and efficiency, yielding the shortest epoch training time (approximately 1h15').

5.3 Experimental Results

To train our VQA model, we implemented the following configurations. At the start of each epoch, the training data were shuffled to prevent the model from learning any unintended patterns from the order of the data. Class weights were computed and normalized to handle the imbalanced dataset, ensuring that the model paid appropriate attention to all classes during training. Several callbacks were employed to enhance the training process: ModelCheckpoint was used to save the best model based on validation loss, EarlyStopping to stop training when the validation loss ceased to improve, and ReduceLROnPlateau to adjust the learning rate dynamically when the validation loss plateaued. Last, the Adam optimizer was chosen for its efficiency and adaptability.

Figure 5.2: Training and Validation accuracy of our VQA model.

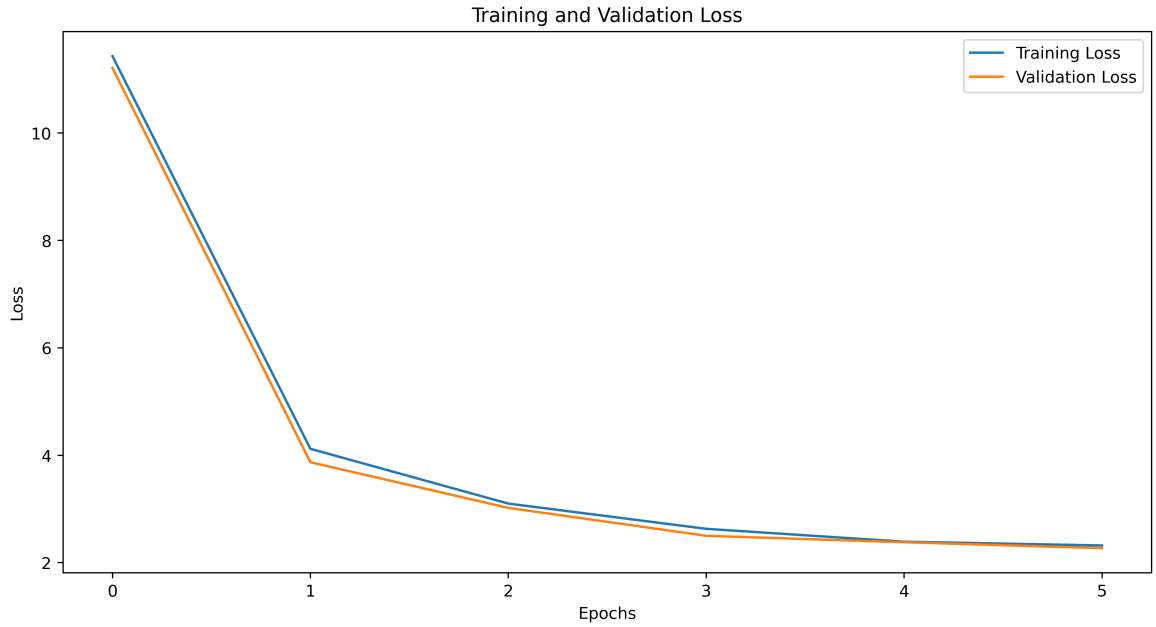
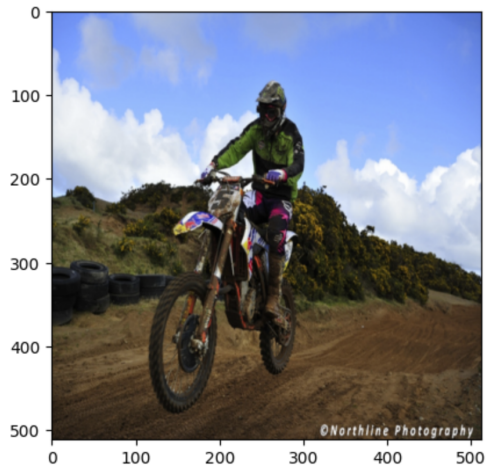


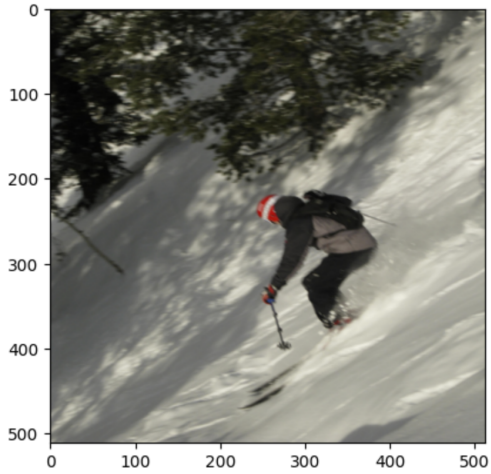
Figure 5.3: Training and Validation loss of our VQA model.

During training, both accuracy and loss were monitored (Figures 5.2, 5.3). Initially, accuracy was low, but increased rapidly during the first two epochs. This sharp rise indicates the model's quick adaptation to the task. As training progressed, the accuracy continued to improve at a slower pace, stabilizing around the fourth and fifth epoch. Regarding the validation accuracy, it slightly exceeded the training accuracy, indicating good generalization of the model without overfitting.

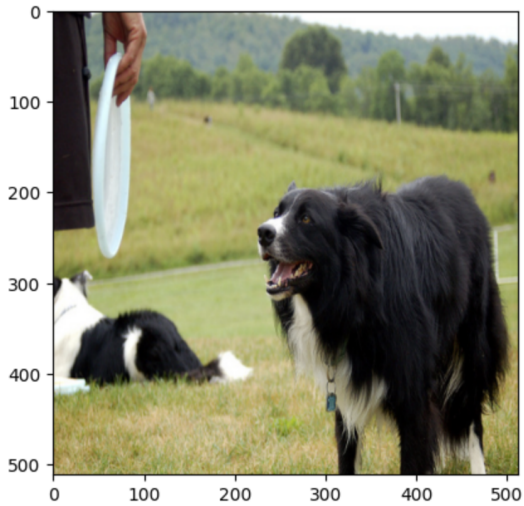
To evaluate the performance of our Custom VQA model, we present a series of results derived from the test dataset (Figure 5.4). These results showcase the model's ability to understand and interpret both the visual content of images and the contextual nuances of corresponding questions. The selected samples vary in difficulty and complexity, providing a comprehensive overview of the model's capabilities. The model seems to recognize different environments, objects and colors within an image. It also effectively understands the relative positions and interactions between objects in a scene.



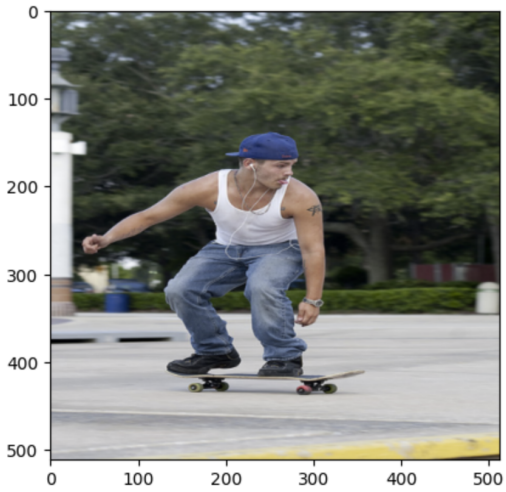
Is the man riding on a dirt path?
prediction: yes
answer: yes



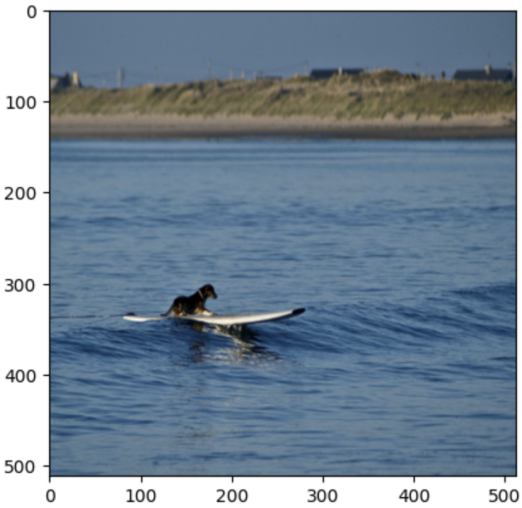
What color is the snow?
prediction: white
answer: white



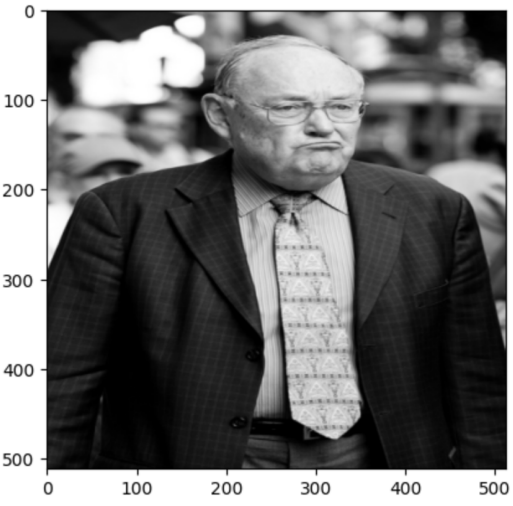
Is the dog waiting?
prediction: yes
answer: yes



How many tattoos can be seen on this man's body?
prediction: 1
answer: 1



Is the dog wearing a collar?
prediction: yes
answer: yes



Is the man smiling?
prediction: no
answer: no

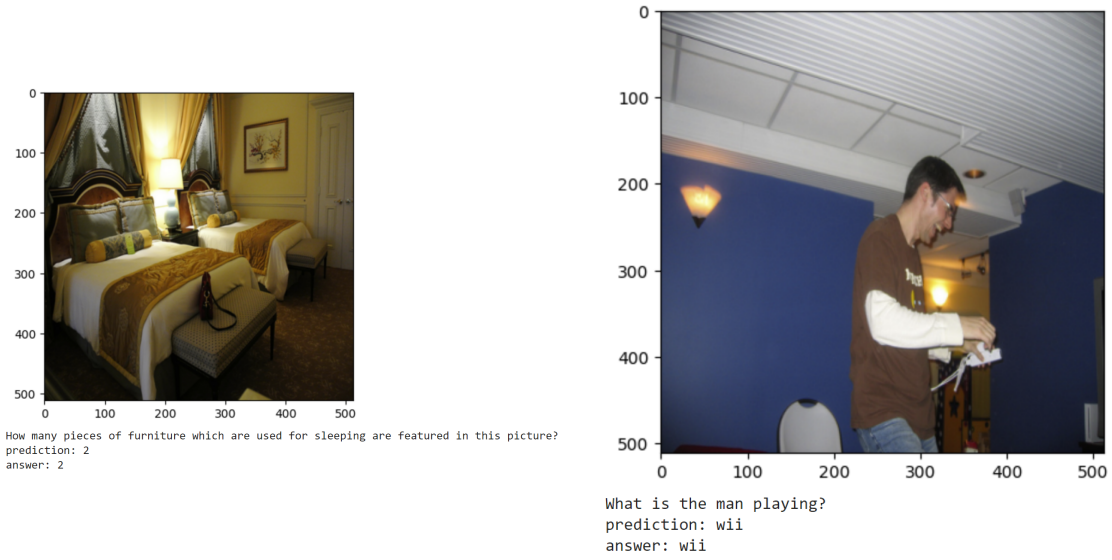
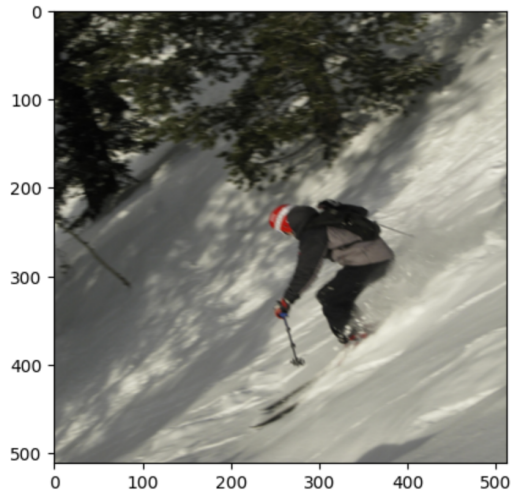


Figure 5.4: Examples of correct predictions made by our VQA model.

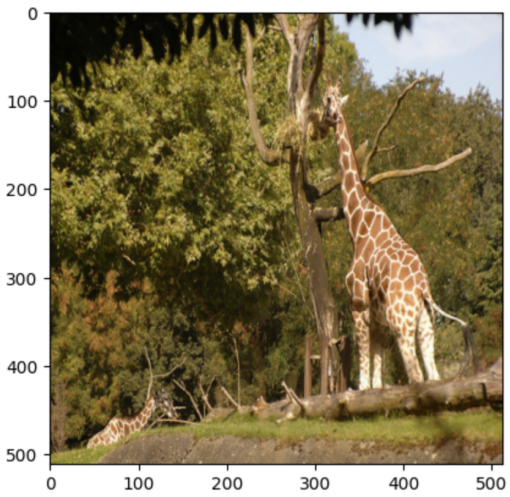
5.4 Error Analysis

In addition to the successful predictions, our Custom VQA model also produced some incorrect answers (Figure 5.5), highlighting areas for potential improvement. These faulty results arise from various challenges inherent to visual question answering tasks.

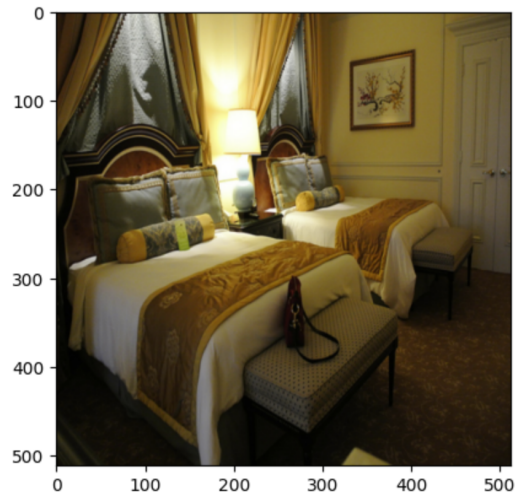
One common issue arises from the complexity of scenes. Images with multiple objects, especially those overlapping or partially hidden, can confuse the model, leading to incorrect answers. Also, we observe that the model can misidentify objects with multiple colors by only predicting the primary color and not its combinations. Moreover, the model might not always grasp the implied meaning intended by the question. These are common challenges inherent to the VQA task. By analyzing these results, we can better understand the limitations of our current model and identify areas for future enhancements.



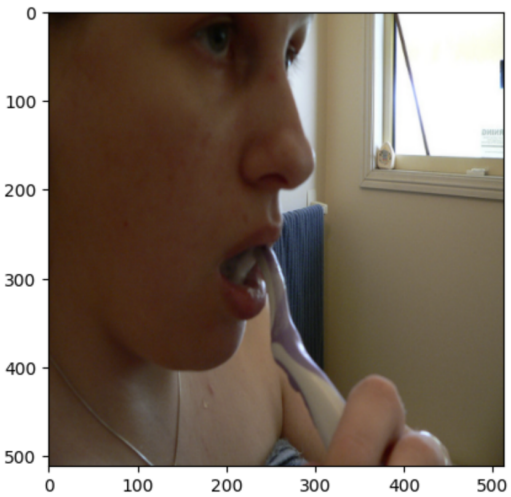
What color is the persons headwear?
prediction: white
answer: red



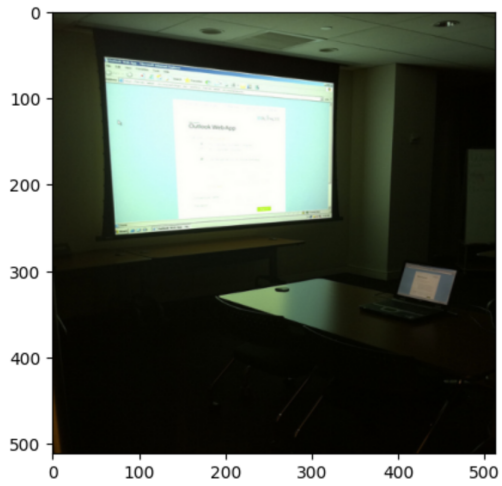
How many giraffes are there?
prediction: 1
answer: 2



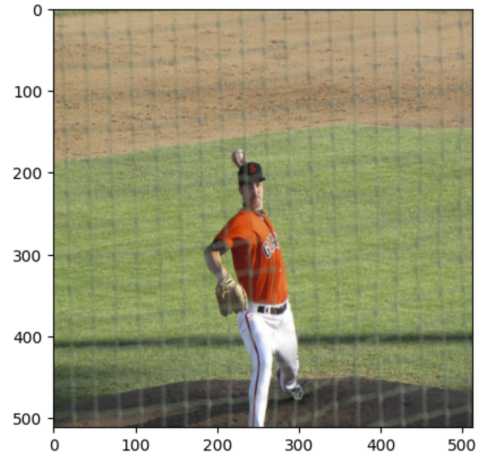
What is sitting on the bench?
prediction: nothing
answer: purse



What color is the toothbrush?
prediction: white
answer: white and purple



What room of the house is this?
prediction: bathroom
answer: office



What position is this man playing?
prediction: baseball
answer: pitcher

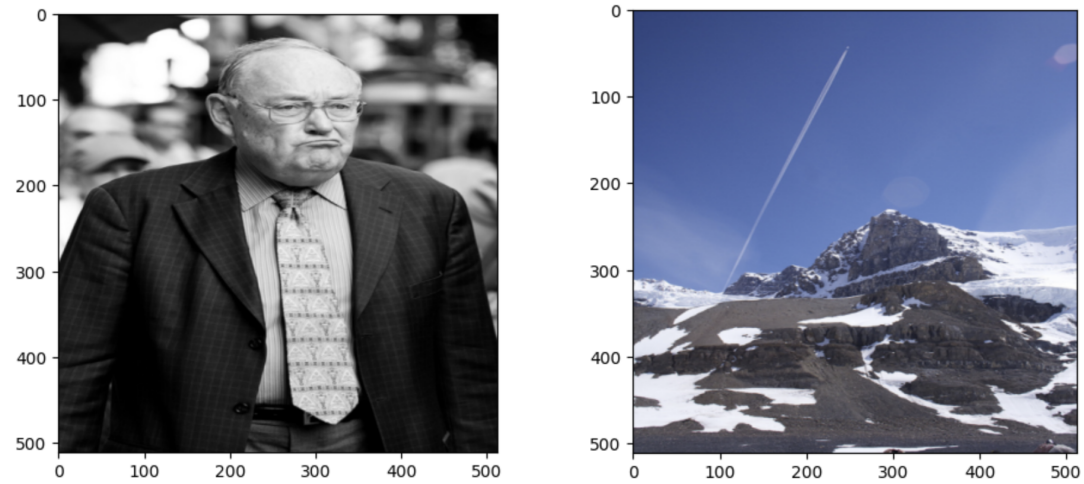


Figure 5.5: Examples of incorrect predictions by our VQA model.

6 Conclusion

Contents

6.1 Limitations

While our custom VQA model has achieved answering correctly many of the testing data questions, it is crucial to recognize its limitations. A significant limitation is the computational resources required by the model for inference and training. The model contains approximately 113.5 million parameters, which impact the feasibility of deploying the model in real-time or resource-constrained environments. Even with access to high-performance GPUs, it is time-consuming and resource-heavy. This aspect restricts the model's applicability in environments with limited computational resources.

Moreover, despite using techniques, like class weights, to address the imbalance in the dataset, the model still exhibits biases towards more frequently occurring answers. This can reduce the reliability of the model's predictions for less common answers.

Finally, the scope of the model's answer classification is limited to the 500 most common answers from the VQA training dataset. This restriction inherently limits

the model's ability to handle a broader range of answers, reducing the potential of a real-world application of visual question answering.

6.2 Future Work

To improve the performance and capabilities of our Custom VQA model several enhancements can be pursued. One significant improvement could be the expansion of the answer vocabulary. Currently, the model is limited to the 500 most common answers from the VQA training dataset. By extending this range, the model will be able to handle a broader spectrum of responses.

Another critical area for future work is enhancing training speed through the use of TPUs (Tensor Processing Units). TPUs are specialized hardware accelerators, designed to expedite machine learning workloads. However, utilizing TPUs requires specific adaptations to the code, such as ensuring compatibility with TPU operations and optimizing data pipelines.

Moreover, by enchancing training speed, it becomes feasible to train larger models and experiment with more complex architectures. One promising architecture would be to use the word embeddings produced by BERT, instead of the single embedding, in order to retain more information from the question. Also we can add an attention layer that would focus on specific parts of the given image based on the question asked.

6.3 Potential Applications

Visual Question Answering (VQA) systems hold immense potential for various real-world applications due to their ability to interpret and respond to questions about visual content. Here are two potential applications of a VQA system:

Autonomous Search and Surveillance Missions

One promising application of a robust VQA system is autonomous search and surveillance operations, particularly when integrated with drone technology. In scenarios such as search and rescue missions, law enforcement drones equipped with cameras can be deployed to cover large areas efficiently. The VQA system processes the images

transmitted by the drone in real time, answering specific queries like "Is there a blue car in the image?" or "Is there a person wearing a red jacket?" This capability is invaluable for identifying specific objects, individuals, or situations within vast datasets of images. If the VQA system detects a positive result, it can instruct the drone to focus on that particular area, allowing for more detailed analysis. A human operator can then review the findings for further action. This application not only enhances the efficiency of search operations but also reduces the need for constant human oversight.

Enhanced Accessibility for the Visually Impaired

Another promising application of VQA technology is enhancing accessibility for visually impaired individuals. A VQA-enabled device, such as a smartphone or smart glasses, can assist users by answering questions about their surroundings. For instance, a visually impaired person could take a picture of a scene and ask, "What objects are in front of me?" or "Is there an obstacle in my path?" The VQA system would analyze the image and provide a descriptive response. This technology can improve the quality of life and independence for those with visual impairments.

References

[1] Yash Goyal et al. "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 (CVPR)*. 2017. URL: https://openaccess.thecvf.com/content_cvpr_2017/html/Goyal_Making_the_v_CVPR_2017_paper.html.

[2] MathWorks. *Deep Learning*. 2024. URL: https://www.mathworks.com/discovery/deep-learning.html.

[3] Siddharth Das. *Computer Vision for Dummies*. Feb. 2020. URL: https://sidereal.medium.com/all-you-need-to-know-about-computer-vision-3997bc6318a6.

[4] Keiron O'Shea and Ryan Nash. "An Introduction to Convolutional Neural Networks". In: *arXiv preprint arXiv:1511.08458* (2015). URL: https://arxiv.org/abs/1511.08458.

[5] JuneHao Ching. *Beginners Guide to Understanding Convolutional Neural Networks*. Sept. 2020. URL: https://pub.towardsai.net/beginners-guide-to-understanding-convolutional-neural-networks-b45386db207b.

[6] Pooja Mahajan. *Max Pooling*. July 2020. URL: https://poojamahajan5131.medium.com/max-pooling-210fc94c4f11.

[7] Kh. Nafizul Haque. *What is Convolutional Neural Network (CNN) in Deep Learning?* Apr. 2023. URL: https://www.linkedin.com/pulse/what-convolutional-neural-network-cnn-deep-learning-nafiz-shahriar.

[8] Robin M. Schmidt. *Recurrent Neural Networks (RNNs): A Gentle Introduction and Overview*. Accessed: 2024-07-26. 2019. URL: https://machinelearningmastery.com/recurrent-neural-networks-rnns/.

[9] Aditi Mittal. *Understanding RNN and LSTM*. Oct. 2019. URL: https://aditi-mittal.medium.com/understanding-rnn-and-lstm-f7cdf6dfc14e.

[10] Paul J. Werbos. "Backpropagation Through Time: What It Does and How to Do It". In: *Proceedings of the IEEE* (1990). URL: https://ieeexplore.ieee.org/document/58337.

[11] Martín Abadi et al. "TensorFlow: A System for Large-Scale Machine Learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (2016). URL: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[12] François Chollet et al. *Keras*. https://keras.io. 2015.

[13] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. https://www.tensorflow.org/. 2015.

[14] Google. *Google Colaboratory (Colab)*. https://colab.research.google.com/. 2020.

[15] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 2019. URL: https://proceedings.mlr.press/v97/tan19a.html.

[16] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2019. URL: https://www.aclweb.org/anthology/N19-1423/.

[17] Peter Anderson et al. "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018. URL: https://openaccess.thecvf.com/content_cvpr_2018/html/Anderson_Bottom-Up_and_Top-Down_CVPR_2018_paper.html.

[18] Ashish Vaswani et al. "Attention Is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*. 2017. URL: https://papers.nips.cc/paper/7181-attention-is-all-you-need.

[19] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *European conference on computer vision*. Springer. 2014. URL: https://cocodataset.org/.

[20] Tashin Ahmed and Noor Sabab. "Classification and understanding of cloud structures via satellite images with EfficientUNet". In: *Earth and Space Science Open Archive* (Sept. 2020). DOI: 10.1002/essoar.10507423.1.

[21] Aishwarya Agrawal et al. "VQA: Visual Question Answering". In: *International Journal of Computer Vision* (2017). URL: https://link.springer.com/article/10.1007/s11263-016-0966-6.

[22] Damien Teney et al. "Tips and Tricks for Visual Question Answering: Learnings from the 2017 Challenge". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2018. URL: https://openaccess.thecvf.com/content_cvprw_2018/html/Teney_Tips_and_Tricks_CVPRW_2018_paper.html.

[23] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations (ICLR)*. 2015. URL: https://openreview.net/forum?id=E5G9rd4IPt.

Appendix

This appendix chapter contains the list of 500 possible answers used by our Custom VQA model. The answers are organized in descending order based on their popularity in the dataset.

- yes
- no
- 1
- 2
- white
- 3
- blue
- red
- black
- 0
- 4
- brown
- green
- yellow
- 5
- gray
- nothing
- right
- frisbee
- baseball
- left
- none
- tennis
- 6
- wood
- orange
- bathroom
- pizza
- pink
- kitchen
- 10
- 7
- cat
- 8
- dog
- water
- man
- skateboarding
- grass
- skiing
- kite
- silver
- black and white
- surfing
- horse
- living room
- skateboard
- phone
- snow
- wii
- giraffe
- woman
- standing
- surfboard
- eating
- cake
- food
- apple
- sunny
- broccoli

- table
- hat
- stop
- purple
- laptop
- elephant
- 12
- sheep
- 9
- snowboarding
- train
- beach
- motorcycle
- soccer
- banana
- bus
- cow
- male
- trees
- walking
- umbrella
- winter
- 20
- wine
- sitting
- flowers
- bear
- camera
- female
- tile
- metal
- clear
- tan
- many
- outside
- brick
- car
- plane
- 11
- sandwich
- donut
- summer
- night
- bananas
- bed
- down
- cloudy
- zebra
- hot dog
- bench
- wall
- cheese
- bird
- fence
- plate
- bedroom
- kites
- tv
- tree
- fork
- people
- helmet
- 15
- sand
- bat
- horses
- beer
- red and white
- skis
- boat
- chair
- birthday
- bike
- street
- cows
- couch
- truck
- tennis racket
- coffee
- up
- floor
- afternoon
- building
- dirt
- glass
- square
- zoo
- suitcase
- 25
- concrete
- airport
- snowboard
- stripes
- nike
- boy
- 13
- teddy bear
- ball
- ocean
- fruit
- usa
- no one
- city
- playing wii
- knife

- round
- girl
- airplane
- english
- plastic
- carrots
- tie
- christmas
- ground
- blonde
- old
- toilet
- flying kite
- blue and white
- oak
- glasses
- open
- scissors
- chinese
- toothbrush
- day
- overcast
- paper
- donuts
- remote
- sidewalk
- africa
- chicken
- fall
- 30
- wii remote
- clock
- cold
- evening
- park
- breakfast
- cooking
- window
- chocolate
- dell
- stone
- mountains
- shirt
- picture
- plaid
- office
- cell phone
- lettuce
- light
- palm
- sun
- vegetables
- apples
- mirror
- 50
- mountain
- field
- clouds
- restaurant
- 14
- rectangle
- elephants
- ski poles
- tomato
- spoon
- solid
- daytime
- nowhere
- person
- sleeping
- carpet
- sunglasses
- sky
- birds
- unknown
- fire hydrant
- computer
- noon
- short
- happy
- counter
- closed
- fish
- wedding
- shorts
- american
- wetsuit
- leaves
- oranges
- desk
- small
- salad
- 16
- house
- on table
- england
- sign
- dinner
- microwave
- home
- lot
- police

- color
- passenger
- backpack
- 100
- luggage
- resting
- milk
- very
- both
- asian
- inside
- carrot
- brown and white
- pepperoni
- morning
- london
- jeans
- zebras
- playing tennis
- circle
- glove
- hair
- pine
- sink
- catcher
- 24
- bowl
- graffiti
- beige
- rainbow
- bag
- towel
- vase
- spinach
- floral
- giraffes
- playing
- books
- safety
- brushing teeth
- hay
- bicycle
- box
- soup
- wii controller
- watch
- gold
- refrigerator
- racket
- rain
- road
- flower
- child
- talking on phone
- spring
- blanket
- jacket
- baby
- on wall
- real
- middle
- bread
- flying
- motorcycles
- china
- windows
- adidas
- neither
- drinking
- wilson
- electric
- background
- on
- in air
- striped
- sneakers
- lights
- rocks
- new york
- 18
- church
- shoes
-
- roses
- bridge
- rainy
- hand
- mouse
- checkered
- smiling
- cars
- young
- gas
- ketchup
- kite flying
- sandals
- playing frisbee
- book
- lunch
- white and blue
- triangle

- umbrellas
- leather
- lab
- in water
- keyboard
- talking
- controller
- pole
- collar
- wire
- texting
- coca cola
- seagull
- batter
- pitcher
- grazing
- tabby
- dessert
- peppers
- heart
- umpire
- shadow
- posing
- goggles
- taking off
- plant
- hot dogs
- stove
- photographer
- sunset
- large
- lamp
- canada
- rope
- good
- clothes
- reading
- rock
- forward
- surfboards
- rug
- double decker
- head
- purse
- lake
- river
- electricity
- train station
- 17
- off
- tennis court
- oven
- tracks
- front
- straight
- bears
- boots
- jumping
- on left
- parking
- soda
- door
- long
- gloves
- meat
- lemon
- pictures
- top
- jet
- clean
- toilet paper
- branch
- shade
- on ground
- olives
- cup
- ceramic
- hotel
- basket
- decoration
- adult
- shelf
- oval
- rose
- new
- taking picture
- water skiing
- red and blue
- ford
- paint
- calm
- indoors
- parking lot
- running
- downhill
- van
- parking meter
- skate park
- warm
- vanilla
- mustard

- delta
- on plate
- north
- cutting board
- flag
- stop sign
- landing
- red and yellow
- modern
- school
- fries
- 40
- boats
- hot
- cargo
- playing baseball
- tomatoes
- bottom
- 200
- rice
- wind
- dining room
- daisy
- smoke
- blinds
- turkey
- beef
- riding
- asphalt
- united
- tea
- red and black