Technische Universität Berlin

Faculty of Electrical Engineering and Computer Science
Dept. of Computer Engineering and Microelectronics
REMOTE SENSING IMAGE ANALYSIS GROUP

Transformer-based Lightweight Visual Question Answering for Earth Observation

Master of Science in Computer Engineering

July, 2022

Hackel, Leonard Wayne

Matriculation Number: 373518

Supervisor:	Prof. Dr. Begüm DEMIR
Advisors:	Kai Norman CLASEN
	Dr. Mahdyar RAVANBAKHSH

Declaration

I hereby confirm to have written the following thesis on my own, not having used any other sources or resources than those listed.

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe angefertigt habe. Sämtliche benutzten Informationsquellen sowie das Gedankengut Dritter wurden im Text als solche kenntlich gemacht und im Literaturverzeichnis angeführt. Die Arbeit wurde bisher nicht veröffentlicht und keiner Prüfungsbehörde vorgelegt.

Berlin, July 22, 2022 ____________________
Leonard Wayne Hackel

Acknowledgements

Firstly, I would like to thank Prof. Begüm Demir, who introduced me to the field of Remote Sensing and Earth Observation.

Secondly, I am very thankful to the staff of the Remote Sensing Image Analysis Group at the TU Berlin for all the help and guidance I received while working on my master thesis. Especially my advisors Dr. Mahdyar Ravanbakhsh and Kai Norman Clasen I would like to thank for the regular consultations and the active support.

Last but not least, I would like to thank my family and friends for always being available to give me their advice and emotional support.

Abstract

In this thesis, a framework for building machine learning networks for solving visual question answering tasks in the domain of Earth Observation is presented. The flexible design of the framework allows the integration of different models for the speech and image domains. Thus, state-of-the-art, transformer architectures can be combined to answer natural language questions using satellite imagery. Also presented is a data set that combines satellite imagery with questions and answers, where unlike other domains, the images have more than the usual three channels of red, green, and blue. Instead, the images include 10 color channels and an additional 2 channels from a synthetic aperture radar.

The validity of the approach is examined through a series of experiments using combinations of four different image processing networks, five text processing networks, and various activation functions, modality combination methods, and dimensions of classification heads. It is shown that while all tested image processing networks can achieve competent results, the choice of text processing network and activation function may have some significant influence on the training results. The choice of modality combination methods and the dimensions of the classification head also have an influence, although it is smaller than in the two previously mentioned categories. Considering the number of parameters, the configuration consisting of MobileViT and $\text{Bert}_{\text{Tiny}}$ as feature extractors, multiplication as feature fusion and 256/128 fusion dimensions is identified as the most promising network of the presented framework with the investigated sub-networks.

Kurzfassung

In dieser Arbeit wird ein Framework zur Erstellung von Machine Learning Netzwerken zur Lösung von Visual Question Answering Aufgaben in der Domaine der Erdbeobachtung vorgestellt. Der flexibel gestalltete Aufbau des Ansatzes erlaubt die Integration verschiedener State-of-the-Art Modelle für die Sprach- und Bilddomaine. Dadurch können aktuelle Transformerarchitekturen miteinander kombiniert werden, um natürlich-sprachlich gestellte Fragen unter Zurhilfenahme von Satellitenbildern zu beantworten. Außerdem wird ein Datensatz vorgestellt, welcher Satellitenbilder mit Fragen und Antworten kombiniert, wobei die Bilder anders als in anderen Domainen über mehr als die üblichen drei Kanäle rot, grün und blau verfügen. Stattdessen beinhalten die Bilder 10 Farbkanäle und zusätzlich 2 weitere Kanäle von einem Radar mit synthetischer Blende.

Die Validität des Ansatzes wird durch eine Testreihe mit Kombinationen aus vier verschiedenen Bilderverarbeitungsnetzwerken, fünf Textverarbeitungsnetzwerken sowie verschiedenen Aktivierungsfunktionen, Modalitätskombinierungsmethoden und Dimensionen von Klassifizierungsköpfen untersucht. Es wird gezeigt, dass zwar alle getesteten Bilderverarbeitungsnetzwerken kompetetive Ergebnisse erreichen können, die Wahl des Textverarbeitungsnetzwerken und der Aktivierungsfunktion aber zum Teil erheblichen Einfluss auf das Trainingsergebnis haben. Die Wahl der Modalitätskombinierungsmethoden und die Dimensionen des Klassifizierungskopfes haben ebenso einen Einfluss, auch wenn dieser geringer ausfällt als in den beiden vorhergenannten Kategorien. Unter Berücksichtigung der Anzahl an Parametern wird die Konfiguration bestehend aus MobileViT und Bert$_{\text{Tiny}}$ als Feature Extraktoren, Multiplikation als Feature Fusion und 256/128 Fusionsdimensionen als vielversprechestes Netzwerk des vorgestellten Frameworks mit den untersuchten Sub-Netzwerken identifiziert.

Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
AUC	Area Under Curve
BigEarthNet-MM	multi-modal BigEarthNet
BigEarthNet-MM-VQA	multi-modal BigEarthNet Visual Question Answering
BERT	Bidirectional Encoder Representations from Transformers
CD	Cloud Discrimination
CLC	CORINE Land Cover
CGA	Cross-Modal Global Attention
CNN	Convolutional Neural Network
CORINE	Coordination of information on the environment
CST	Cross-Modal Spatial Transformer
CSWin	Cross-Shaped Window
DL	Deep Learning
DNN	Deep Neural Network
DY-ReLU	Dynamic ReLU
ELU	Exponential Linear Unit
EO	Earth Observation
EOVQA	Visual Question Answering for Earth Observation
FCM	Feature Combination Method
FFM	Feature Fusion Method
FFN	Feed Forward Network
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GAN	Generative Adversarial Networks
GB	Gigabyte
GELU	Gaussian Error Linear Unit
GRU	Gated Recurrent Unit
HR	High Resolution
HSI	Hyper-Spectral Imagery

IB-BERT Inverted-Bottleneck BERT

IW Interferometric Wide Swath

KD Knowledge Distillation

LePE Locally-enhanced Positional Encoding

LCC Land Cover Classification

LiT Locked-image tuning

LiT-4-EOVQA Lightweight Transformer-based Visual Question Answering Framework for Earth Observation

LLM Large Language Model

LN Layer Norm

LR Low Resolution

LSTM Long Short-Term Memory

LU Land Use

mAP Mean Average Precision

MCB Multimodal Compact Bilinear Pooling

ML Machine Learning

MLM Masked Language Modeling

MLP Multi-Layer Perceptron

MM Multi-Modal

MSA Multi-Head Self Attention

MSE Mean Squared Error

MSI Multi-Spectral Imagery

MUTAN Multimodal Tucker Fusion for Visual Question Answering

NAS Neural Architecture Search

NIR Near Infrared Light Spectrum

NLP Natural Language Processing

NSP Next Sentence Prediction

OOD out-of-distribution

PE Positional Encoding

QA Question Answering

QFormer Question Feature Extraction Sub-Network

ReLU Rectified Linear Unit

RGB Red, Green, Blue

RNN Recurrent Neural Network

ROC Receiver Operating Characteristic

RS Remote Sensing

RSVQA Visual Question Answering for Remote Sensing

SAR Synthetic Aperture Radar

SOP Sentence Order Prediction

SPCL Selfpaced Curriculum Learning

SVM Support Vector Machine

SWIR Short-Wave Infrared Light Spectrum

TB Terabyte

TN True Negative

TP True Positive

TPR True Positive Rate

UAV Unmanned Aerial Vehicle

VFormer Visual Feature Extraction Sub-Network

VIS Visible Light Spectrum

ViT Vision Transformer

VQA Visual Question Answering

1 Introduction

The ever-increasing amount of data that is recorded every day by a wide variety of sensors is an ever-present problem. For example, in February 2020, over 500 Gigabyte (GB) of video data were uploaded to the video platform youtube.com per minute [1]. In other domains such as Earth Observation (EO) a lot of data is generated too. For example, ESA's satellite-based Copernicus Sentinel-1, -2 and -3 missions produce more than 20 Terabyte (TB) of data per day [2].

The data generated by satellite missions can be used by everyday tasks such as weather forecasting [3, 4], as well as in specialized fields such as climate change analysis [5, 6], urban planning [7, 8], Land Cover Classification (LCC) and Land Use (LU), surface change [9–12], or disaster relief and prevention [13–15]. Even very specialized problems such as drainage tile detection [16, 17] or crop insurance fraud [18] can be solved by remote sensing imagery.

However, pure, unprocessed data cannot be meaningfully interpreted by humans, but require pre-processing such as filtering or structure, for example through visualization. This turns data into information that can be assimilated and further processed or applied by humans.

The processing of this amount of data is no longer possible by humans, because too much data is recorded. Therefore, automated systems have to be developed, which allow to filter and process the amount of produced data quickly and effectively with the help of computers. To ensure that the data do not lose their relevance or become distorted during processing, these systems must work reliably and quickly. Due to the amount of data to be processed, the systems should also operate efficiently to reduce the energy load as well as the computing power requirements of the processing systems.

In addition, flexible systems that can run on devices with low computational power are needed for use cases such as disaster relief. For example, Deep Learning (DL) models can be used in flood disasters to identify flooded areas and support relief efforts. These can be based on different modalities such as satellite imagery [15, 19] or posts in social media [20]. However, finding the structure, i.e. the architecture, of such a system is a complex undertaking, since the behavior of the networks depends on the exact structure, which in turn depends on many design decisions. This is especially true for Artificial Neural Networks (ANNs), since in these systems the parameters are learned by training with large data sets rather than set by design choices.

Approaches such as Neural Architecture Search (NAS) are supposed to help solve this problem by automating the search for appropriate architectures [21], but they are criticized for consuming large amounts of energy during the search itself. Even though the amounts of energy used are sometimes overestimated many times over [22], it is clear that energy efficiency is an important part of the development of new architectures.

The low computational demands of lightweight and throughput-optimized networks would allow them to run on mobile devices such as laptops. This means that the networks can be applied directly at the site of the disaster and the results of the execution can be implemented directly and immediately on site. This eliminates the need to wait for access to large data centers, which would require a stable network connection and possibly the allocation of computing resources.

Current DL models like transformer and Convolutional Neural Networks (CNNs) tend to have a lot of parameters. This means that the models require a lot of computational power for both training and application. In addition, these large models no longer fit into the memory of a single machine and must be trained using strategies such as distributed learning. However, such strategies require large data centers with sometimes hundreds of nodes and scale only conditionally or not at all depending on the chosen strategy [23, 24].

In order to simplify access to Machine Learning (ML) models for non-domain persons, special model families are developed. VQA models are one such approach, that allows the processing of image data in conjunction with a natural language input. This makes it possible, for example, for first responders in disaster areas to use ML to simplify their work without the user needing a computer science background.

The goal of this work is to develop a VQA model that has a small number of parameters and computational requirements compared to state-of-the-art models. For this purpose, already existing lightweight architectures based on transformers are to serve as a basis. The Thesis consists of 6 chapters which are structured as follows.

Section 2 presents related work to this thesis. A short overview over transformers in image processing domains not related to earth observation and the current state of the art of DL models in remote sensing image applications in general will be given. The chapter is opened with the explanation of various efficient training methods for ANNs. Additionally, a detailed description of all subnetworks used in pre-training and training as well as work related to VQA is provided.

Section 3 will explain the overall structure and functionality of the visual question answering model introduced in this thesis. The individual stages as well as the overall concept and the motivation behind the individual design decisions will be discussed.

Section 4 presents the multi-class-multi-label data set used used for pre-training of the Visual Transformer models. It will also introduce the created data set for Multi-Modal (MM) Visual Question Answering for Earth Observation (EOVQA).

Section 5 will introduce the methodology used to test the framework proposed in the previous chapter in practice. The results of the training are presented.

In section 6, the results are evaluated and overall conclusions are drawn. In addition, the implications of the results are presented and discussed. The work in this thesis is summarized and a final result drawn from the individual partial results. In addition, possible future work will be presented in this section.

2 Related Work

This chapter describes the current developments and fundamental principles of DL in general and in the Remote Sensing (RS) field in particular. It also explains the basics of efficient learning, VQA and presents influential work related to multi modal learning.

2.1 Overview on Efficient Learning

Efficient training is fundamentally important in an age where ever more data needs to be processed faster. In efficient training, one must distinguish between two types of efficiency: time efficiency and space efficiency.

Space efficiency is about minimizing the memory footprint of certain objects, in this case networks. This means that a space-efficient network, compared to a less efficient network, consumes less memory both during backup and during use in the main memory of the computer or in the graphics memory of the graphics card or other accelerator hardware.

One possible method to reduce the memory footprint is the *weight sharing* method introduced by Nowlan [25]. This method, which does not train each layer weights individually but uses e.g. weight matrices multiple times, reduces the footprint of the network by using less individual weights. Weight sharing has already been successfully applied several times in different variations [26–28].

Another method of removing the number of used parameters is *pruning*. In this method, parameters that have either no influence or only a negligible influence on the final result are removed from the network. This means that these parameters do not have to be stored and their (non-existent) influence does not have to be calculated during use, which reduces the memory and computing power requirements of the network. Although the correct application of pruning is still under discussion, the method has been applied frequently with varying degrees of success [29–32].

Quantization is a network compression and acceleration method in which the weights of the individual layers are partially or completely transferred from one data type to another. Normally weights are stored as high precision floating point data, but it has been shown that converting to fixpoint with lower bit widths can increase the speed and memory footprint of networks with small losses in accuracy as shown for the conversion from float32 to 16-bit or smaller fixed point by Lin et al. [33]. Alternatively, conversion to 8-bit integer [34] or even reduction to only one or two bits per weight and activation [35] has been successfully demonstrated.

Instead of replacing the weights of the network after pre-training and before finetuning with another representation, as is the case with quantization, you can also use parts of the network and the optimizer during training instead of float32, float16. This conversion can be done either manually [36] or with an approach called *automatic mixed precision* by frameworks. This can mitigate the problems caused by simply replacing 32-bit floats with 16-bit floats [37]. It was also shown that the approach is very scalable to increase the training speed on clusters with few as well as many accelerators [38].

Also methods like *frozen layer* or *zero/few-shot learning* can be counted as efficient methods to a limited extent. With frozen layers, only parts of the model, typically the last n layers, are trained, while all other layers keep their current state. Zero-shot learning, on the other hand, trains a text embedding and an image embedding simultaneously, where the actual test classes are not explicitly included in the training set. During the test time, the text embedding closest to the image embedding is used as the classification. This allows the model to abstract to unseen, or in the case of Few-Shot learning, to little seen classes. More details are discussed in section 2.4.

These two methods are efficient in that they greatly speed up training. In the aspects of memory usage and execution time during test time or after deployment, however, they have no advantage over the same architecture that has not been trained with frozen layer or zero/few-shot.

Knowledge Distillation Originally proposed as a methodology by Hinton et al. [39], Knowledge Distillation (KD) is based on the thesis that a large network (the “Teacher” network) contains many parameters that do not have much influence on the overall result. Therefore, a small network (the “Student” network) can be trained by the larger one to incooperate the same knowledge in fewer parameters. In language processing in the context of this thesis, these large networks are, for example, Large Language Models (LLMs).

The goal of Distillation can be formulated as the minimization of the distillation loss L_{KD} with

$$L_{KD} = \sum_{x \in X} L\left(f^S(x), f^T(x)\right) \tag{1}$$

where X is the data set, f^S and f^T are behavior functions, that describe the transformation of the input of a network (f^S for the student network and f^T for the teacher network) to any information and L is a loss, that describes the difference between the student and teacher network. As a basic case this can be the cross entropy loss between the logits of the two networks. However, the behavior function can also describe the transformation to any intermediate layer of the teacher network.

For single class classification, the classification head usually calculates the probability q_i of class i by comparing the network logit z_i of this class with all logits using the softmax function

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{2}$$

Here T is the temperature coefficient, usually defined as 1. For $T > 1$ this creates softer labels, meaning, the ground-truth labels are treated as less certain.

The basic form of KD trains one or more teacher models on a large data set until these models have achieved satisfactory performance. Then, the student model is trained by using the soft labels that the teacher models output instead of the hard original label.

Alternatively, the labels can be adjusted for distillation by either combining the hard labels weighted with the soft labels, or by using two loss functions in parallel. Thus, a cross-entropy loss with high temperature in the softmax calculation with target of the soft labels and a second cross-entropy loss with $T = 1$ with target of the hard labels train simultaneously on soft and hard labels. The total loss is calculated as the weighted sum of the two cross-entropy loss functions.

This method can also be used if only for a part of the training data the hard labels are known. In these cases the weighting for the hard labels is set to zero for data where hard labels are unknown. Meanwhile, care must be taken that the gradients scale with the factor $1/T^2$, so they should be multiplied by T^2 to keep the relative influences of hard and soft labels constant for T during the hyperparameter search. In general, a strong focus on the soft labels seems to work better for knowledge transfer from teacher to student model.

2.2 Overview on Transformers

The Transformer Architecture is an architecture of ANNs specifically designed for sequential data processing. It was proposed by Vaswani et al. [40] and is considered a successor architecture to Long Short-Term Memorys (LSTMs) [41] and Recurrent Neural Networks (RNNs), especially gated RNNs [42]. Typical application areas for sequential data are mainly Natural Language Processing (NLP)-tasks like language modeling or machine translation.

Recurrent models process the input in series. For each symbol of the input at position t a hidden state h_t is calculated from the input and the hidden state h_{t-1}. This serial processing prevents parallelization and makes the calculation slow. This is especially relevant for long input sequences. The Transformer model, on the other hand, enables the parallelized processing of sequential data.

Instead of recurrence, the Transformer architecture mainly uses the attention principle. This mechanism allows the network to make connections between distant and close symbols in the same calculation step. These connections are the relationships of symbols to each other. Unlike recurrent models, these relationships can be computed in a highly parallelized manner, which greatly reduces the execution time of the networks.

Attention can be modeled as

$$\text{Attention}(q, K, V) = \sum_{i=1}^{N} \text{softmatch}_a(q, K)_i \cdot v_i \tag{3}$$

$$\text{softmatch}_a(q, K)_i = \frac{\exp(a(q, k_i))}{\sum_{j=1}^{N} \exp(a(q, k_j))} = \text{softmax}_i(\{a(q, k_j\}_j) \tag{4}$$

with $q \in Q \subseteq \mathbb{R}^{d_q}$ a query in query-space, $k \in K \subseteq \mathbb{R}^{d_k}$ a key in key-space and $V \subseteq \mathbb{R}^{d_v}$ the value-space. a is an alignment function $Q \times V \to \mathbb{R}$ which evaluates how well the embeddings at positions i and j relate to each other [43]. This function can be realized e.g. as dot product ("dot-product attention") or as Feed Forward Network (FFN) ("additive attention").

Transformers use dot-product attention, which can be parallelized by processing all queries and keys in parallel as a matrices Q and K instead of serially as a vector:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(QK^T\right) V \tag{5}$$

Since the softmax function produces only a small gradient for large values, the result of the matrix multiplication is usually scaled depending on the dimensionality of the key embedding d_k:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{6}$$

Other architectures have already been able to use attention successfully. Lin et al. [44] used attention to create the embedding of the input. Instead of a vector, the embedding was represented as a 2D-matrix, where each row corresponds to a symbol of the input. This allowed successful author profiling, sentiment classification and textual entailment.

The Transformer architecture uses an Encoder-Decoder structure. The tokens (embedding of the symbols) of the input $\mathbf{X} = \{x_1, \dots, x_n\}$ are transformed by the Encoder into a representation $\mathbf{Z} = \{z_1, \dots, z_n\}$. From this representation, the output $\mathbf{Y} = \{y_1, \dots, y_m\}$ is auto-regressively generated. The Decoder receives the representation $\mathbf{Y}$ and all outputs $\{y_1, \dots, y_t\}$ generated up to $t < m$ as input to generate the output symbol y_{t+1}.

Both encoder and decoder consist only of self-attention and fully-connected layers as learnable parameters. These two layers together with Layer Norm (LN) and residual connections form the building block of each transformer block as shown in fig. 1.

The encoder blocks are L identically constructed blocks, each consisting of two sub-blocks. The first sub-block is a Multi-Head Self Attention (MSA) layer, followed by LN. The second block consists of a fully-connected layer, also followed by LN. A residual connection is connected around each of the two sub-blocks.

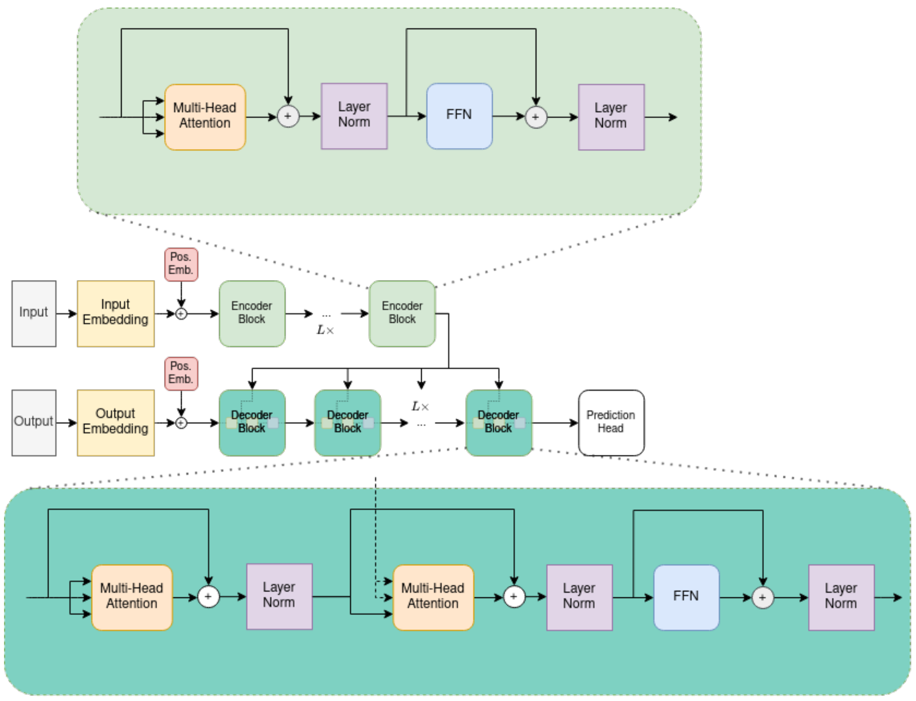


Figure 1: Structure of the Transformer Architecture [40, fig. 1]

Therefore, each of the sub-blocks produces embeddings of the same dimensionality d_H.

Multi-head Attention does not calculate the attention based on the dimension of the embedding, but projects this embedding first linearly into the dimensions of the query d_q, key d_k and value d_v. This projection is learned and is different for each head. The heads then calculate the attention in parallel. All results are concatenated and again projected linearly back into the initial dimension d_H.

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \dots, \mathrm{head}_i)W^O \tag{7}$$

$$\mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{8}$$

with $W_i^Q \in \mathbb{R}^{d_H \times d_q}$, $W_i^K \in \mathbb{R}^{d_H \times d_k}$, $W_i^V \in \mathbb{R}^{d_H \times d_v}$ and $W^O \in \mathbb{R}^{Ad_v \times d_H}$ where A is the number of parallel heads.

Since the attention mechanism is inherently position invariant, the transformer network must be given additional position information for each token. This Positional Encoding (PE) is attached to or in-cooperated with the input embedding. The embedding can be generated in different ways, e.g. learned or as proposed by Vaswani et al. [40] by sinuide functions.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_H}) \tag{9}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_H}) \tag{10}$$

The decoder also consists of L identical blocks, each of which consists of both sub-blocks contained in the encoder and an additional third sub-block. Between the MSA and the fully-connected layer is an additional multi-head cross-attention layer, which computes the relation between the final embedding of the encoder-stack and the embeddings of the decoder. Here, the queries are the output of the previous self-attention sub-block and keys and values are the embedding of the encoder-stack. This block contains layer-norm and a residual-connection as well.

In addition, the self-attention block is modified by masking all positions $i \geq t$ for decoding of position t. This prevents the self-attention block from attending to positions that have not been decoded yet.

2.2.1 The Evolved Transformer

The development of the Transformer architecture led to other architectures based on the Transformer principle. For example, So et al. [21] introduce the Evolved Transformer. This group of architectures uses the tournament selection evolutionary algorithm as described by Goldberg and Deb [45]. This algorithm works as follows:

The set of all parameters that describe the configuration of a network is called a gene encoding. The NAS called algorithm aims to search the high-dimensional search space of all possible genes encodings as efficiently as possible to find a gene encoding that describes a high-performance network.

For this purpose, an initial generation of networks is created whose genes are randomly initialized. All networks are then evaluated (trained and tested). The two best networks form the parents for the next generation. The genes of the parents are mixed and mutated (noise is added) to create a new generation of networks. All lower performing networks are discarded. This procedure is repeated for a certain number of steps or until a satisfactory performance of the networks is achieved.

Although the networks found at NAS were more performant than previously used networks, they were soon overtaken by architectures found by hand. In addition, NAS is considered computationally expensive and polluting, even though the impact of the search is limited by techniques such as Progressive Dynamic Hurdles [21] and advances in hardware or datacenter efficiency [46].

2.2.2 BERT

A very influential architecture is the BERT architecture introduced by Devlin et al. [47]. This architecture differs from the original transformer architecture by using only the encoder-stack of the architecture presented by Vaswani et al. [40] as shown in fig. 2.

In addition, two pre-training techniques are introduced that allow the encoder to be universally pre-trained in an unsupervised fashion and fine-tuned only for

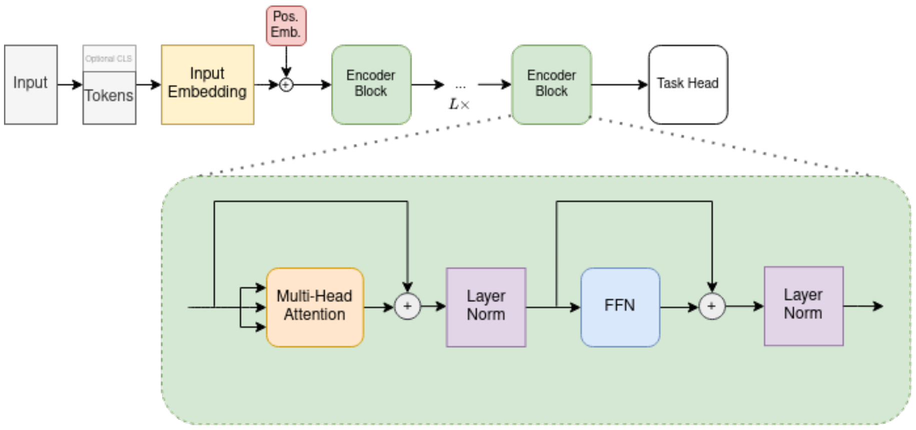


Figure 2: Structure of the BERT Architecture

the specific task, requiring a smaller amount of labeled data for fine-tuning. The two tasks are Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

Both tasks rely on the network architecture being a bidirectional encoder. This means, that in the attention heads the token at position i can attend to the token at position j with $i < j$ and vise-versa.

In contrast to ViT, BERT uses two additional special tokens. During embedding, an additional `[CLS]` token is prepended to the sequence of token. This token was introduced for classification tasks. Instead of feeding all final hidden states $T_i \in \mathbb{R}^H$ with $i = 1 \dots N$ as input into the classification head, only the features of the `[CLS]` token (denoted as $C \in \mathbb{R}^H$) are used as input. Additionally, a `[SEP]` token was introduced. This token allows to use multiple sentences as one sequence as an input to the network. Sentences are divided using the `[SEP]` token.

In addition, two learned embeddings have been introduced, which indicate for each input token whether it belongs to sentence A or sentence B. Including the position embedding as used in ViT and the individual token embedding, the full embedding structure of a pair of sentences is build as shown in fig. 3.

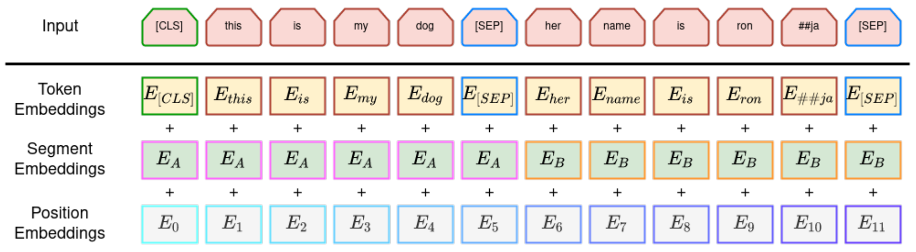

Input
[CLS]
this
is
my
dog
[SEP]
her
name
is
ron
##ja
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{this} E_{is} E_{my} E_{dog} $E_{[SEP]}$ E_{her} E_{name} E_{is} E_{ron} $E_{\#\#ja}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10} E_{11}

Figure 3: Embedding generation for example sentence pair "This is my dog. Her name is Ronja." [47, fig. 2]

Masked Language Modeling is based on the Cloze Procedure [48]. Its working principle is, that a part of the input tokens (15% in the case of BERT) are masked at random. This means, that instead of the token embedding, a special `[MASK]` token is used. The last hidden state of the masked tokens are passed through an output softmax layer over the full vocabulary to predict the masked word.

However, this `[MASK]` token has the disadvantage that it is not present in most fine-tuning tasks. Therefore, the inputs for pre-training and fine-tuning are fundamentally different. To reduce this difference, only 80% of the masked words are actually replaced by the `[MASK]` token. In the other cases, the tokens are replaced with equal probability by random embedding or the corresponding embedding.

For the evaluation of the token at position i, not all hidden states are tested, but only the hidden state T_i using cross entropy loss. It does not matter whether the masked token was replaced by `[MASK]`, a random token or not at all. For MLM all tokens use sentence A embedding, as only a single sentence is used as input.

Next Sentence Prediction is a classification task, which requires that the network can extract the relationship between two sentences. This is especially relevant for tasks like Question Answering (QA), where the answer to the textual question is provided in a text excerpt[1]. For training, two sentences are used as an input with embeddings created as shown in fig. 3. In 50% of all cases, sentence B is the immediate next sentence of sentence A, whereas in the other cases it is a different, random sentence from a corpus.

For binary classification, the final hidden vector C of the input token `[CLS]` is evaluated to predict if sentence B is the subsequent sentence of sentence A. Therefore, this token must have all features of the inter-sentence relationship encoded for correct classification.

2.2.3 Efficient BERT-like Architectures

In the following paragraphs, some efficient transformer architectures are described. All architectures have in common that their structure is a variation of the BERT architecture and that they use optimized training procedures to train the architecture efficiently and with only small restrictions in the final ability of text understanding despite a small number of parameters. All approaches are using the BookCorpus data set [49] and English Wikipedia for pre-training of the model. The differences in training procedure and architecture will be described in the following paragraphs.

The models described will be used for feature extraction of the NLP-input of the framework described in section 3 of this thesis.

[1]as opposed to VQA, where the answer to the textual question is provided in a provided image

DistilBERT Sanh et al. [50] introduce a smaller version of BERT [47], with the number of layers $L = 6$, the hidden size $H = 768$, the feed forward/filter size $I = 3072$ and the head number $A = 12$, which is the same as architecture as BERT$_{\text{Base}}$, but using only half as many layers. Pre-trained weights are obtained from Huggingface [51]. This model is about 40% smaller and 60% faster than BERT$_{\text{Base}}$ while retaining 97% of the performance.

To train this compressed model, a triple loss was introduced, which conditions the model not only on the language itself, but also learns intermediate representations of a larger teacher model. This trains a kind of behavioral cloning (see section 2.1). The three loss functions are a language modeling loss as used for example for MLM L_{MLM}, a distillation loss L_{CE} and a cosine-distance loss L_{cos}. The cosine-distance loss is used to bring the hidden states of the teacher and student models closer together. It is calculated as

$$L_{cos}(x, y) = \begin{cases} -\cos(x_1, x_2) & \text{if } y = 1 \\ \max(0, \cos(x_1, x_2)) & \text{if } y = -1 \end{cases} \tag{11}$$

where $x = \{x_1, x_2\}$ are the two vectors whose distance is to be measured and $y = \{-1, 1\}$ is the label.

The distillation loss is defined as

$$L_{CE} = \sum_i t_i * \log(s_i) \tag{12}$$

where t_i and s_i are the probability estimations of the teacher and student model respectively.

For the initialization of the weights, every second layer of the BERT teacher model was copied in each case. This is possible because only the number of layers differs from the training model.

It could be shown that this kind of initialization increases the performance of the final network compared to a random initialization. Additionally, using all three loss functions improves over using only a subset of the loss functions.

TinyBERT The model introduced by Jiao et al. [52] follows the overall structure of BERT [47], but with the number of layers $L = 4$, the hidden size $H = 312$, the feed forward/filter size $I = 1200$ and the head number $A = 12$ for the specific configuration. Pre-trained weights as used in this Thesis are made available by Huggingface [53] used in this thesis. It leverages KD by introducing three loss functions for different parts of the model. The loss functions are motivated by the findings of Clark et al. [54], who discovered, that the attention heads contain linguistic properties like syntax or coreference.

For distillation a two step approach is presented. In the first step, a mapping $n = g(m)$ is defined, which maps the index of layer $g(m)$ of the teacher network to the index of layer m of the student network. The embedding layer is set to index

0 and the prediction head to layer $M + 1$. The layers are therefore layer 0 and $g(0)$ for the embedding and $N + 1 = g(M + 1)$ for the prediction layer.

In the second step, a distillation loss for each layer of the student model is calculated over the whole data stet. The distillation loss of layer m is defined as

$$L_{\text{model}} = \sum_{x \in X} \sum_{m=0}^{M+1} L_{\text{layer}} \left(f_m^S(x), f_{g(m)}^L(x) \right) \tag{13}$$

Here, L_{layer} is a loss function that is dependent on the type of layer that is distilled.

For the embedding layer, the loss function uses the Mean Squared Error (MSE) between the embeddings, defined as

$$L_{\text{embd}} = \text{MSE}(E^S W_e, E^T) \tag{14}$$

with E^S and E^T being the embeddings of student and teacher respectively and $W_e \in \mathbb{R}^{d' \times d}$ being a learnable linear transformation from student embedding space to the teacher embedding space.

For the transformer blocks, the loss is composed of two parts. For the multi head attention blocks, the loss is

$$L_{\text{attn}} = \frac{1}{h} \sum_{i=1}^{h} \text{MSE}(A_i^S, A_i^T) \tag{15}$$

with h being the number of attention heads and $A_i \in \mathbb{R}^{l \times l}$ the attention matrix for head i with l being the length of the token sequence. Note, that A_i is used directly instead of the attention output softmax(A_i)

BERT$_{\text{Tiny}}$ The models introduced by Turc et al. [55] are following the exact structure as proposed by Devlin et al. [47]. The specific model used in this thesis is the configuration called BERT$_{\text{Tiny}}$ (the smallest configuration) with pre-trained weights used from Huggingface [56], with the number of layers $L = 2$, the hidden size $H = 128$, the feed forward/filter size $I = 512$ and the head number $A = 2$.

This model was trained together with 23 other configurations (each differing by a combination of different L and H[2]) in a three-stage procedure. Only a small amount of labeled data is necessary to apply this three-stage procedure. However, a large amount of unlabeled data is also required for training.

In order to use this procedure, a teacher model with a high performance was trained beforehand (see section 2.1). This model does not have to meet any deployment speed/size constraints, since it is only needed for training the efficient model and is not used in the final application. In the first step, the student network is trained with in an unsupervised manner. Specifically, the network used here was trained with MLM (see section 2.2.2). In the second step, the knowledge

[2] A and I are calculated as $A = H/64$ and $I = 4H$ for all models

of the larger teacher is distilled as described in section 2.1. The data used for distillation may overlap with the labeled data, but this is not necessary. As a final, optional step, the student model can be fine tuned using the labeled data set. The model used in this thesis was pre-trained using a BERT$_{\text{LARGE}}$ teacher model for the distillation (second step).

MobileBERT MobileBERT is a model architecture introduced by Sun et al. [57] that makes modifications to the BERT architecture specific to mobile devices to make the resulting models smaller and faster. The used modifications reduce the width rather than the depth, because this leads to better performance, as Turc et al. [55] have shown. To achieve this, the output dimensionality of both the attention head and the FFN is reduced.

In order to preserve the residual connection and the depth-independent dimension of the embedding, a dimension reducing linear layer is in parallel to the multi head self attention and an expanding linear layer after the FFN as can be seen in fig. 4b.

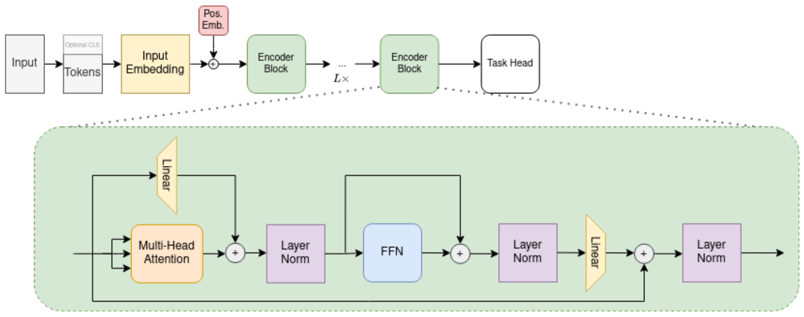
For the training of this network, an additional training objective was defined, which is used during training. It transfers knowledge from individual levels of the trainer to the corresponding level of the student. Since the embedding of a transformer accounts for a significant part of the overall network size, its width should also be reduced, but the dimensions of the attention head output as well as the FFN should be as in BERT [47], the teacher network was modified similarly to the student, except that the expanding and contracting linear layers are reversed as seen in fig. 4a. A detailed dimension comparison is listed in table 8 in the Appendix.

The additional training objective L^l_{KT} is to transfer knowledge from layer l of the teacher to layer l of the student. The loss is linearly composed of two parts. First, the knowledge of the attention maps is to be transferred using an attention transfer method. This is done by reducing the KL-divergence between the individual heads of the teacher and the student.

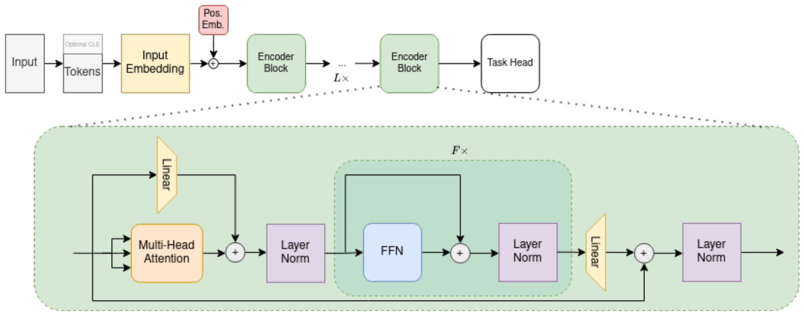
$$L^l_{\text{AT}} = \frac{1}{TA} \sum_{t=1}^{T} \sum_{a=1}^{A} D_{\text{KL}} \left(a^{tr}_{t,l,a} || a^{st}_{t,l,a} \right) \tag{16}$$

Here, l is the index of the layer, T the sequence length and A the number of the attention head.

Since the in-/output dimensions of each layer of the teacher match those of the student, a comparison can be made between the feature maps at the end of each layer. The mean squared error between the teacher and student feature maps is the second part of the training objective. It is described by



(a) Architecture of the MobileBert teacher network: Inverted-Bottleneck BERT (IB-BERT) introduces an inverted bottleneck expanding parallel to the MSA layer and dimension reducing after the FFN + Add & Norm block [57, fig. 1b].



(b) Architecture of the MobileBert student network: Same general structure as the teacher network but using a bottleneck instead of an inverted bottleneck and repeating the FFN + Add & Norm block F times [57, fig. 1c].

Figure 4: MobileBert teacher and student network: Knowledge is transferred using Attention Transfer of MSA heads and Feature Map Transfer after the last LN of the teacher to the student for each Encoder Block.

$$L^{l}_{\text{FMT}} = \frac{1}{TN}\sum_{t=1}^{T}\sum_{n=1}^{N}\left(H^{tr}_{t,l,n} - H^{st}_{t,l,n}\right)^2 \tag{17}$$

with N being the feature map size.

Through this loss function, the layer by layer knowledge of the teacher is transferred to the student. Sun et al. [57] have shown that splitting the loss in practice into normalized feature map discrepancy and feature map statistics discrepancy can stabilize the training.

Furthermore, a pre-training distillation loss L_{PD} is defined utilizing MLM, KD and NSP in a linear combination.

$$L_{\text{PD}} = \alpha L_{\text{MLM}} + (1-\alpha)L_{\text{KD}} + L_{\text{NSP}} \tag{18}$$

with $\alpha \in [0, 1]$ being a hyperparameter.

Three different training approaches were proposed. In the first approach, all layers are trained simultaneously, all loss functions are applied as a linear combination and all layers are trained simultaneously. This method is called Auxiliary Knowledge Transfer.

The second approach works in two steps. In the first step, the student model is pre-trained by the transfer loss L_{KT}. In the second step, the classification head and all layers are finetuned by the pre-training distillation loss L_{PD}. The approach is named Joint Knowledge Transfer.

The third approach, called Progressive Knowledge Transfer, works in N steps when N is the number of layers (encoder blocks and classifier head) of the model. In the first step, only the first encoder block is trained by knowledge transfer. The embedding is not trained, but copied from the teacher and already frozen for this and following steps.

In the second step, the first block is frozen and only the second block is trained. In the third step, the first and second blocks are frozen and only the third block is trained, and so on.

In the last step, instead of training using L_{KT}, the classification head is copied from the teacher and then finetuned. In this step all encoder blocks and the embedding are also fine tuned, there is no more frozen layer. A variation of the method used in practice is instead of freezing the layer, to train it with a very small learning rate. In practice, this method yielded the best results. Pre-Trained Weights are taken from Huggingface [58].

ALBERT The BERT-like architecture developed by Lan et al. [59] attempts to scale the size of networks without being blocked by poor hardware scaling. As Vaswani et al. [40] have been able to show, deeper and wider networks generally have better performance than smaller networks. However, beyond a certain size, these networks no longer fit into the memory of a single accelerator such as a GPU or TPU, requiring the use of multiple accelerators. This leads to communication overhead, which can be avoided by reducing the parameters in the network used.

The Albert architecture has made two modifications to the BERT architecture to reduce the number of parameters. Firstly, embedding is factorized and secondly, parameters are shared across different layers. In addition, a new training objective (inter-sentence coherence loss) is introduced that predicts the order of two input sentences.

Factorized embedding parameterization solves the problem that the hidden size H of the network is bound to the embedding size E ($H \equiv E$). In the embedding layer the embedding is mapped linearly from the vocabulary size V to the embedding size E using a linear layer, so that the fully connected layer has the size $O(V \times H)$. Especially for a very large vocabulary this mapping is very parameter intensive. Factorized embedding parameterization splits this embedding into two step layers, with $H \not\equiv E$. In the first layer V is mapped to a low dimensional embedding space E and in the second to the hidden space H.

This changes the total embedding size to $O(V \times E + E \times H)$, which can be significantly smaller if $H \gg E$. In addition, the dimensionality of H is no longer bound to the dimensionality of the embedding E.

The second optimization is split weights between the layers. Either the attention head weights, or the FFN weights, or both can be shared. The different weight-sharing strategies were tested with the result that especially FFN-sharing reduces the performance of the network. However, these weights also make up a significant portion of the network, with some saving more than 50% of all weights through this strategy.

Splitting into groups was also tested. Here M blocks share weights with $M < L$. For example, the network with $L = 12$ could be divided into 3 groups with $M = 4$ each, so that the number of parameters is approximately divided into thirds. In general, it was found that less weight sharing leads to better performance, but massively increases the number of parameters. The Albert network therefore shares all weights, so that the number of parameters in the network does not scale with depth, but depends only on embedding size E and hidden size H[3]. Thus, the configuration ALBERT$_{\text{BASE}}$ with $L = 12$, $H = 768$ and $E = 128$ used in this thesis with pre-trained weights used from Huggingface [60] has about 12M parameters, whereas the configuration BERT$_{\text{BASE}}$ with $L = 12$ and $H = E = 768$ used by factorized embedding parameterization has about 108M parameters. However, the training time still depends on the depth and deeper networks generally perform better.

The training objective inter-sentence coherence loss introduced in connection with ALBERT replaces the unsupervised NSP loss. While MLM learns the coherence between words, NSP is supposed to learn the coherence between sentences. However, Lan et al. [59] argue that NSP predicts the subject of the two sentences rather than learning the coherence.

The inter-sentence coherence loss is intended to remedy this problem, in that here the training samples consist only of positive samples of NSP. However, in 50% of the cases, the order of the two segments are swapped. The network should use the Sentence Order Prediction (SOP) task to determine whether the given order of the segments is correct or swapped. Thus, only coherence is learned, since the topic of the segments is probably the same. It could be shown that NSP cannot solve SOP, while SOP can nevertheless solve NSP with slight losses. This leads to an increased performance of downstream tasks, which contain multiple sentences as input.

[3]and thus also the number of attention heads $A = H/64$ and feed forward/filter size $I = 4H$

2.2.4 Visual Transformers

DL approaches are increasingly used in vision tasks [61, 62]. DL is also being used with increasing frequency in EO applications. For example, State-of-the-Art architectures like CNNs, (stacked) Autoencoders, RNNs or Generative Adversarial Networkss (GANs) are used in remote sensing applications like LCC, search-and-retrieval [63] or domain adaptation [64, 65].

Due to the success of transformers in NLP applications, it is reasonable to conclude that the concept can also be applied to the vision domain. The problem here is that the input data, in this case the image, must be converted into a sequence and the computational complexity of the transformer scales quadratically with the sequence length. If one would simply flatten the image by considering each pixel individually as 3 tokens, one per channel, one would already have a sequence length of 3072 for a relatively low resolution image with a resolution of $32 \times 32 \times 3$ as in the CIFAR-10 and CIFAR-100 data sets [66].

To overcome this limitation, optimization approaches such as local attention [67], attention applied to specific regions [68] or only one axis [69, 70] were used. Also sparcity [71] or the use of patches of size 2×2 [72] was tried. However, all these approaches had little success or were not scalable enough.

Kuznetsova et al. [73] suggest that the image is embedded with much larger patches. They propose the sizes 14×14 or 16×16 pixel. This would leave a single image from the CIFAR data sets with a sequence length of only 4 and standard image sizes such as 256×256 and 224×224 with sequence lengths of only 256 or 196 for a patch size of 16. A patch size of 14 is only possible for images of size 224×224, as 256 is not integer dividable by 14 and not used in these cases. 224×224 images with patch size 14 have a sequence length of 256.

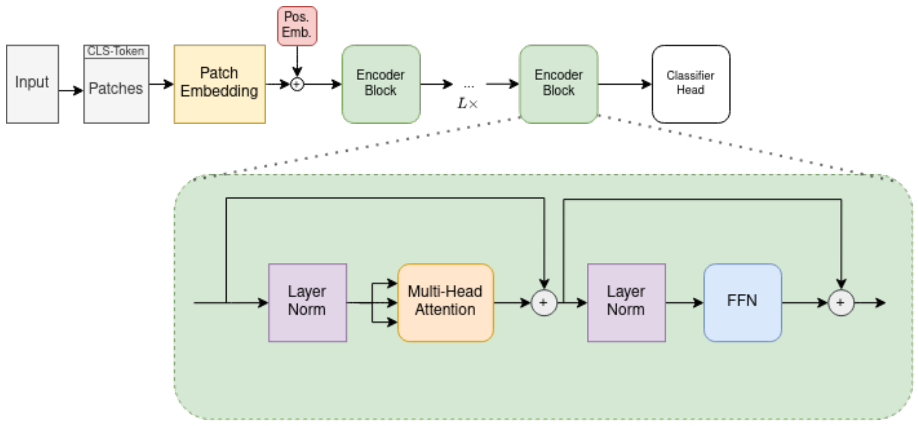


Figure 5: Structure of the ViT Architecture [73, fig. 1]

An image $x \in \mathbb{R}^{H \times W \times C}$ is divided into a sequence of non-overlapping patches $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where each patch has a resolution of $P \times P$. The number of patches is dependent on the image and patch resolution and calculated as $N = HW/P^2$.

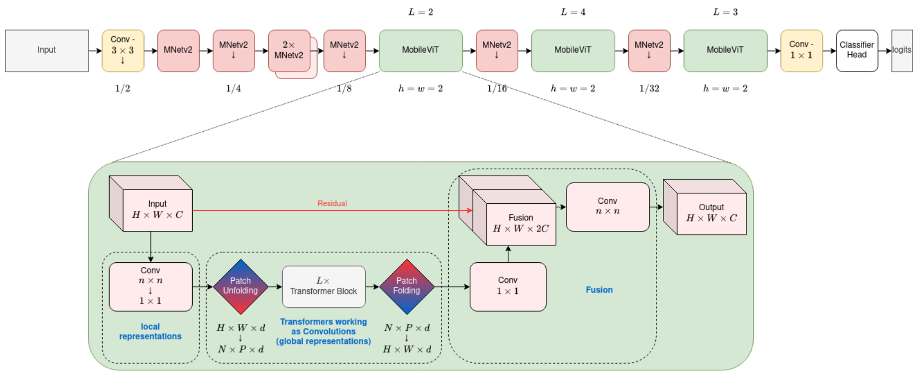


Figure 6: Structure of the MobileViT network [76, fig. 1b]

Like in the BERT architecture [47] (see section 2.2.2), a `[CLS]` token is prepended, thereby the input sequence to the transformer has the length $N + 1$. For classification, only the `[CLS]` token is feed into a classifier head.

The tokens are projected into embeddings using a patch embedding projection E, that projects the flattened patches into a latent space of dimension d_H, which is also the hidden dimension throughout the full network. E can be a linear layer or a feature map from a CNN, thereby building a hybrid architecture, or any projection satisfying the dimension requirements. Positional embeddings are added to the patch embeddings. The sequence is then used as input to the transformer architecture.

Unlike in the BERT architecture, the LN is before the Attention layer and the Multi-Layer Perceptron (MLP) as shown in fig. 5. ViT is thus defined by

$$z_0 = [x_{[CLS]}, x_p^1 E, x_p^2 E, \ldots, x_p^N E] + E_{pos} \qquad E \in \mathbb{R}^{(P^2 \cdot C) \times D}, E_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{19}$$

$$z'_l = \text{Attention}(\text{LN}(z_{l-1})) + z_{l-1} \qquad l = 1 \ldots L \tag{20}$$

$$z_l = \text{MLP}(\text{LN}(z'_l)) + z_{l-1} \qquad l = 1 \ldots L \tag{21}$$

$$y = \text{LN}(z_L^0) \tag{22}$$

Training this architecture requires a large amount of data due to the lack of inductive bias present in CNNs. Therefore, the networks used must be complemented by data sets such as ImageNet-21K [74] or the JFT-300 data set [75] used by google inhouse before they can be finetuned on other data sets. To solve this problem and the long training time of 230 - 2500 TPUv3-core-days depending on the data set and the size of the network, there are a number of optimization proposals. The different optimized architectures used in this thesis are discussed in the following paragraphs.

MobileViT Mehta and Rastegari [76] introduce the MobileViT architecture. It is a hybrid of Transformer blocks and MobileNet-V2 blocks introduced by Sandler et al. [77]. The underlying concept of MobileViT is the assumption that a mixture of convolutions and transformer blocks can simultaneously learn the local, image-specific, spatial inductive bias and learn global relations.

For this, MobileViT-blocks are introduced. They aim to project the local and global information of a given input into an output with fewer parameters. Local spatial information are extracted from the input features by a $n \times n$-convolution and projected via linear combination to a d-dimensional space via point-wise[4]-convolution. These two steps transform the input $\mathbf{X}$ from $\mathbb{R}^{H \times W \times C} \to \mathbb{R}^{H \times W \times d}$ with $d > C$, thereby increasing the dimensionality of the image.

The features are flattened into non-overlapping patches $\mathbf{X}_U \in \mathbb{R}^{P \times N \times d}$ with $P = wh$ pixels in a patch, $N = \frac{WH}{P}$ as number of patches and $h \leq n$ and $w \leq n$ as patch height and width respectively. Here the size of the patches is at most as large as the kernel of the previous filter.

Relations of the patches $p \in \{1, \dots, P\}$ can be modeled by:

$$\mathbf{X}_G(p) = \text{Transformer}\left(\mathbf{X}_U(p)\right), 1 \leq p \leq P \tag{23}$$

By folding the patches to the original image height and width dimensions $\mathbf{X}_G \in \mathbb{R}^{P \times N \times d} \to \mathbf{X}_F \in \mathbb{R}^{H \times W \times d}$ and re-projection into a lower C-dimensional space, the original dimensions $H \times W \times C$ are restored. This tensor can encode both local (from $\mathbf{X}_U(p)$) and global (from $\mathbf{X}_U(p)$) information of every other pixel. The tensor is concatenated with the input $\mathbf{X}$ and low level and high level features combined via $n \times n$-convolution, thereby creating an output tensor of the same size as the input tensor.

This novel MobileViT-block is combined with standard MobileNet-V2 and down-sampling by factor of 2 MobileNet-V2-blocks as shown in fig. 6 before being feed into a final point-wise convolution and a classifier head. The configuration used in this Thesis is MobileViT-XXS.

MobileFormer The MobileFormer architecture introduced by Chen et al. [78] is a parallel structure of two networks, interconnected via two-way bridges. It combines advantages of the local processing capabilities of MobileNet-V2 inverted bottleneck blocks introduced by Sandler et al. [77] and standard Transformer blocks leveraging attention and FFN as used by Kuznetsova et al. [73]. Additionally, local and global features are combined after every step via information exchange bridges modeled as cross attention as shown in fig. 7.

The reduction in computational complexity while not reducing the performance was the main goal in the creation of this network. This is the motivation behind multiple design decisions.

In contrast to the original ViT [73], this network does not use projected patches of images as input for the Transformer blocks. Instead, randomly initialized tokens $\mathbf{Z} \in \mathbb{R}^{N \times d}$ are used. Thereby, the number of patches N is not dependent

[4] 1×1

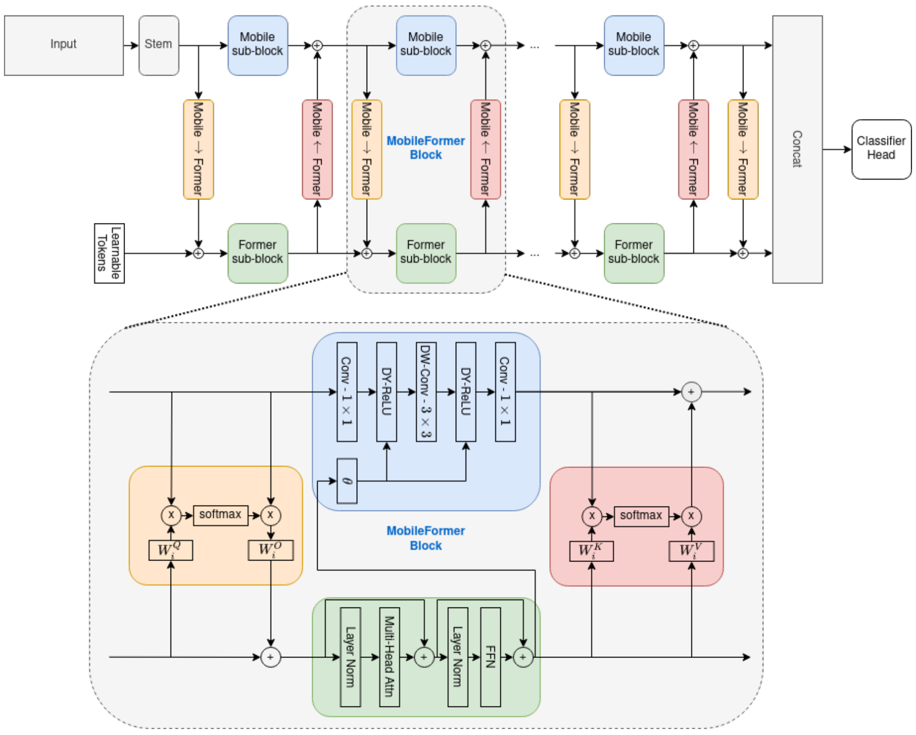


Figure 7: Structure of the MobileFormer network [78, fig. 1, 3]

on the size of the input image $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$ but can be chosen as desired. This allows the Transformer path to reduce the complexity by lowering the required quadratic calculation in the multi-head self attention.

Local and global features are fused using cross attention in the Mobile → Former and Mobile ← Former blocks. However, in contrast to other implementations of cross attention, this block does not use projections (W_i^Q, W_i^K and W_i^V) at the Mobile part but only at the Former part. This reduces the computational cost of this part of the network, as the number of tokens in $\mathbf{Z}$ is lower than the number of positions in the local features $\mathbf{X}_L$. Additionally, the local input features $\mathbf{X}_L$ to both blocks are taken from the bottleneck of the Mobile path, where the number of channels is low.

Local and global features are split into individual maps $\mathbf{X}_L = [\mathbf{x}_h]$ and $\mathbf{Z} = [\mathbf{z}_h]$ for heads $h \in \{1, \dots, H\}$ in multi-head attention. Local to global features in the Mobile → Former block are then calculated as

$$\mathbf{z}_{\text{out}} = \mathbf{z} + \Big[\text{Attention}(\mathbf{z}_h W_h^Q, \mathbf{x}_h, \mathbf{x}_h)\Big]_{h=1:H} W^O \tag{24}$$

where W^O combines the output of multiple heads into the same dimension as the input.

Similarly, global to local features in the Mobile $\leftarrow$ Former block are modeled as

$$\mathbf{x}_{\text{out}} = \mathbf{z} + \left[\text{Attention}(\mathbf{x}_h, \mathbf{z}_h W_h^K, \mathbf{z}_h W_h^V)\right]_{h=1:H} \tag{25}$$

W_h^Q, W_h^K and W_h^V are the projection matrices for Query, Key and Value for the global features in head h respectively.

In between the two bridge blocks are the Mobile block for local features and the Former block for global attention. The Former block uses standard multi-head self attention with a FFN afterwards. However, to reduce computational complexity of the FFN, the expansion ratio was reduced from 4 to 2. Additionally, layer normalization was used as suggested in [79].

For the Mobile block, inverted bottlenecks block as proposed in [77] are used. However, as activation, Dynamic ReLU (DY-ReLU) [80] instead of ReLU is used, using the output of the Former block as input for the hyper parameter θ. Finally, features from the Mobile and Former path are concatenated and feed into a classifier head. The configuration used in this Thesis is MobileFormer-52.

CSWin The Cross-Shaped Window (CSWin) Transformer architecture proposed by Dong et al. [81] is based on multiple Transformer blocks, where groups of blocks are separated by a downsampling convolution each. The CSWin Transformer blocks use Cross-Shaped Self-Attention instead of regular Self-Attention. Additionally, instead of inducing positional encoding at the embedding step as done by Kuznetsova et al. [73], Locally-enhanced Positional Encoding (LePE), is used. LePE induces local positional information at every attention step.

CSWin uses a convolution 7×7, stride 4, as Token Embedding. This produces input features $\mathbf{X} \in \mathbb{R}^{H/4 \times W/4 \times C}$. However, no positional information is added to the tokens at this step. Instead, the tokens are the input to CSWin Tranformer blocks. These use the general architecture of Transformers as presented by Kuznetsova et al. [73] with layer normalization in front of the FFN and the Self-Attention as proposed in [79].

Dong et al. [81] propose two improvements to the Transformer architecture: LePE and Cross-Shaped Self-Attention.

Cross-Shaped Self-Attention Instead of regular Self-Attention, the heads are divided into Vertical and Horizontal Self-Attention and a split head before feed into multiple attention heads. For Horizontal Self-Attention, the input features are split into non-overlapping strips of $\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^M]$ where each strip is of equal width sw and each strip containing $sw \times W$ tokens. Horizontal Self-Attention can be modeled as

$$\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^M] \tag{26}$$

$$Y_k^i = \text{Attention}(\mathbf{x}^i W_k^Q, \mathbf{x}^i W_k^K, \mathbf{x}^i W_k^V) \tag{27}$$

$$\text{H-Attention}_k(\mathbf{X}) = [Y_k^1, Y_k^2, \dots, Y_k^M] \tag{28}$$

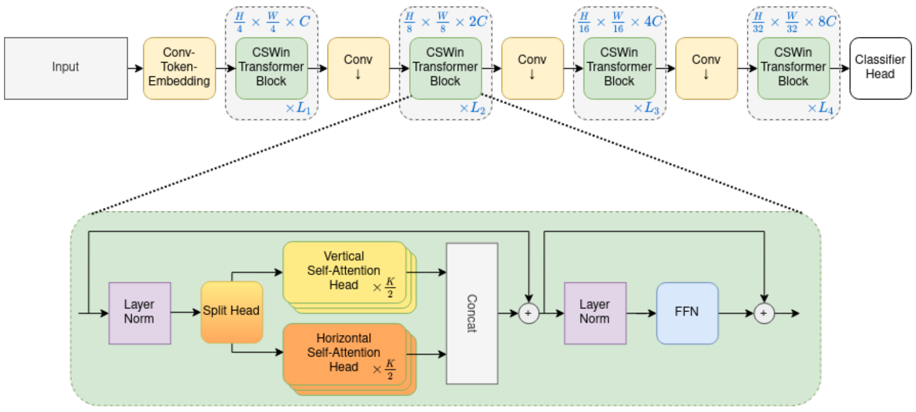

Input
Conv-Token-Embedding
CSWin Transformer Block
$\frac{H}{4} \times \frac{W}{4} \times C$
$\times L_1$
Conv ↓
$\frac{H}{8} \times \frac{W}{8} \times 2C$
$\times L_2$
$\frac{H}{16} \times \frac{W}{16} \times 4C$
$\times L_3$
$\frac{H}{32} \times \frac{W}{32} \times 8C$
$\times L_4$
Classifier Head
Layer Norm
Split Head
Vertical Self-Attention Head $\times \frac{K}{2}$
Horizontal Self-Attention Head $\times \frac{K}{2}$
Concat
FFN

Figure 8: Structure of the CSWin network [81, fig. 1]

where $\mathbf{x}^i \in \mathbb{R}^{sw \times W \times C}$, $M = H/sw$ and $i = 1, 2, \dots, M$. $W_k^Q, W_k^K, W_k^V \in \mathbb{R}^{C \times d_k}$ are the projection matrices for query, key and value for head k with $d_k = C/K$ respectively with K being the number of heads. Vertical Self-Attention can be derived similarly.

For a total of K attention heads, $K/2$ heads are each allocated to Horizontal and Vertical Self-Attention. Cross-Shaped Attention is the projection of the concatenation of all K heads.

$$\text{CSWin-Attention}(\mathbf{X}) = \text{Concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_K)W^O \tag{29}$$

$$\text{head}_k = \begin{cases} \text{H-Attention}_k(\mathbf{X}) & k = 1, 2, \dots, K/2 \\ \text{V-Attention}_k(\mathbf{X}) & k = K/2+1, K/2+2, \dots, K \end{cases} \tag{30}$$

with $W^O \in \mathbb{R}^{C \times C}$ being the projection matrix to project the patch dimension into the output dimension. The attention module is followed by a FFN like the the original Vision Transformer [73].

The network contains $i = 4$ groups of L_i CSWin Transformer blocks. After each but the last group the features are feed into a downscaling 3×3, stride 2 convolution that reduces the feature height and width by 2 while doubling the number of channels as shown in fig. 8.

The standard division of the receptive field (strip width sw) is set to (1, 2, 7, 7) for image size 224×224, therefore had to be changed for image size 128×128[5] to (1, 2, 1, 7) to make the feature map size divisible by the strip size. Except for this change, the network used in this thesis is the CSWin-T configuration.

LePE Instead of encoding positional information while creating the token embedding, LePE induces positional information at every Self-Attention step. LePE introduces a channel-wise bias on a sequence $\mathbf{x} = (x_1, \dots, x_n)$ of length n with a

[5]used image size in this thesis

corresponding output sequence $\mathbf{z} = (z_1, \dots z_n)$. Self-Attention can be calculated as

$$z_i = \sum_{j=1}^{n} \alpha_{ij} v_j \tag{31}$$

with

$$\alpha_{ij} = \exp(q_i^T k_j / \sqrt{d}) \tag{32}$$

with q_i, k_i and v_i being the query, key and value obtained via linear tranformation of the input x_i and d being the feature dimension. LePE can be modeled as

$$z_i^k = \sum_{j=1}^{n} (\alpha_{ij}^k + \beta_{ij}^k) v_j^k \tag{33}$$

for z_i^k being the kth element of vector z_i. The bias β_{ij}^k is set to 0 if the Chebyshev distance $\delta = |j - i|$ is larger than a threshold τ ($\tau = 3$ for the network used in this Thesis). This reduces computational complexity for long sequences which means large images in this case.

ConvMixer The ConvMixer architecture introduced by Trockman and Kolter [82] is a CNN-like architecture which uses token embeddings like other Transformer architectures. Instead of decreasing the spacial resolution while increasing the dimensionality like other CNNs, this architecture follows the principal of Transformers by keeping the dimensions of the image the same throughout the full network.

ConvMixer uses the approach of token embedding as presented by Kuznetsova et al. [73]. However, instead of using a transformer architecture for feature extraction, mixing of tokens as proposed by Tolstikhin et al. [83] for the MLP-Mixer architecture is used. For ConvMixer mixing via MLPs was exchanged for using convolutions instead. Every ConvMixer block uses one depth-wise convolution followed by a point-wise convolution as shown in fig. 9. This combination allows for mixing of features in spatial and channel dimensions.

Each convolution is followed by an activation, specifically Gaussian Error Linear Unit (GELU) for the architecture used in this thesis, and batch normalization. The token embedding is modeled as convolution $\mathbb{R}^{H \times W \times C} \to \mathbb{R}^{H/P \times W/P \times h}$ with H, W, C, P and h being the image height, width, channels, the number of patches and the dimension of the embedding respectively, followed by an activation and batch normalization as well:

$$\mathbf{x}_0 = \mathrm{BN}(\sigma\{\mathrm{Conv}_{C \to h}(\mathbf{X}, \texttt{stride} = P, \texttt{kernels\_size} = P)\}) \tag{34}$$

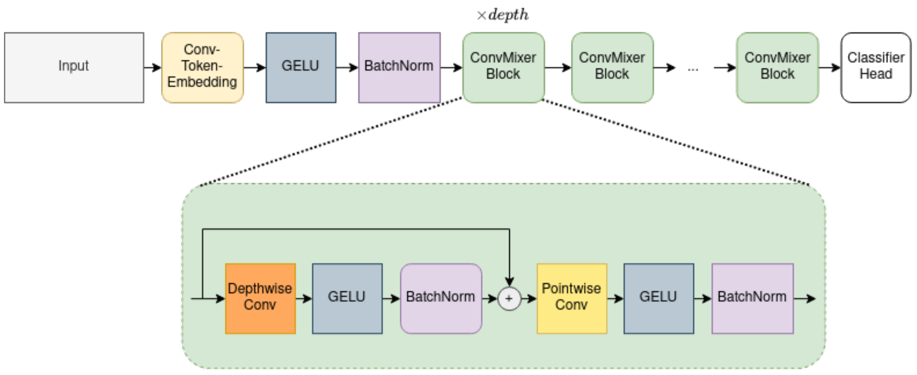


Figure 9: Structure of the ConvMixer network [82, fig. 2]

ConvMixer block i can be modeled as

$$x_i' = \mathrm{BN}(\sigma\{\mathrm{ConvDepthwise}(\mathbf{x}_i)\}) + \mathbf{x}_i \tag{35}$$

$$\mathbf{x}_{i+1} = \mathrm{BN}(\sigma\{\mathrm{ConvPointwise}(x_i')\}) \tag{36}$$

To compensate for the absence of MLPs and attention, which enable the global receptive field in transformer architectures, the kernels of the depth-wise convolutions for ConvMixer were chosen very large when compared with other CNN-architectures.

The configuration 768_32[6] was used for this theses. Here, the embedding dimension is $h = 768$ and the depth of the network is $d = 32$. This model was chosen, as this was designated as the model with the highest throughput.

2.3 Vision and Language Modeling

Visual Question Answering (VQA) is the task to answer a question with the help of a picture in the context of this picture. While some questions can be answered with relatively high certainty by knowledge alone (“What color is the sky?” $\rightarrow$ “Blue”), other questions need a context to be answered (“What color is the car?”). Therefore, it is necessary to develop systems that can combine multiple modalities (e.g. text + image or text + video) to solve this task. Especially the answering of detailed questions requires a wide range of capabilities from classification to object and activity recognition as well as different reasoning techniques [85].

In order to combine the two modalities of text and image, different approaches are being developed simultaneously. Generative models such as DALL-E [86], DALL-E 2 [87], CLIP [88] and Imagen [89] have attracted particular attention. These learn by training a text encoder and a vision encoder in parallel to keep the latent space of the two encodings as close to each other as possible. Thus, an

[6]Some implementations [84] use Rectified Linear Unit (ReLU) instead of GELU for this configuration. This thesis uses the suggested GELU-activation.

image can be reproduced from the latent space of the text encoder or a text can be completed. For example, the latent space can become a classifier by giving as visual input the image and as text input something like "An image of a `[MASK]`." and interpreting the replacement of the `[MASK]` token as a classification.

With this method, these models can be used as zero-shot classifiers or as image-text retrieval, but by training the two encoders separately, the system lacks the ability of reasoning. They only align the latent space without actually combining the features of the two inputs. Therefore, these networks are not suitable for tasks like VQA.

2.4 Overview on Visual Question Answering

Li et al. [90] suggest using a pre-trained region proposal system such as Faster-RCNN [91] along with a text as input to a BERT encoder. The embedding for the text is composed of the embedding of the token, the position of the token and an embedding for the text segment. The segment embedding here indicates that it is a text token and additionally, if there are multiple text inputs, to which text input it belongs, similar to pre-training by NSP in BERT models.

The image embedding is composed of a visual feature representation, a bounding box embedding and a segment embedding. The visual feature representation comes from the region proposal system, as does the bounding box embedding. This is only supplied if the task requires it. The segment embedding indicates that this embedding is part of an image (as opposed to part of a text).

The network is pre-trained with two tasks similar to the structure of the BERT pre-training. First, a variation of MLM is used, where the region of the image belonging to the masked word is not masked. Second, sentence-image prediction is used, a training technique in which the network is given two descriptions of the image and the image and is asked to determine whether it is both descriptions of the image. Here, one of the text inputs always describes the image and the other has a 50% chance of not matching the image, while in the other case it is an alternative description.

Through both pre-training tasks, the BERT encoder learns to combine text and vision features. This can be used in a later finetuning step to train e.g. a VQA network on a VQA data set. It helps to pre-train the network task-specific by training MLM with image on the target data set before the actual supervised training using input, output and objective of the data set.

The flamingo model introduced by Alayrac et al. [92] takes multiple images as input as well as special formatting of the text. The formatting with special tokens `[EOC]` (End of Chunk) and `[image]` allows the model to take multiple images with multiple associated texts as input simultaneously. Perceiver resamplers [93] together with frozen vision encoders produce a fixed number of image features, independent of the number of vision inputs. A stack of language modeling blocks combined with cross attention blocks combines the text inputs with the vision features.

This allows the system to solve tasks based on examples. For example, a VQA task can be solved using as first input an image with text input in the form of "<question>? Answer: <answer>" and as second input an image with text input "<question>? Answer:" is given as the second input. The network then completes the input with the answer to the second question in the context of the second image.

The CoCa architecture proposed by Yu et al. [94] uses contrastive loss as latent space alignment model like CLIP use, but only as part of the training procedure. The CoCa model uses an encoder-decoder structure, where the decoder is split into a unimodal and a multimodal text decoder. In the unimodal text decoder, no cross attention of latent space of the vision encoder is used. Instead, the `[CLASS]` tokens of the unimodal text and vision encoders are trained by a contrastive loss. Here the decoder still works as an encoder. The vision encoder was previously trained e.g. by classification.

After pre-training the vision encoder and aligning the latent spaces, the unimodal text decoder is used as a text generator and the multimodal text decoder incooperates the vision features of the vision encoder. The goal here is to exactly reproduce the caption of the image. Since a lot of attention has to be paid to image details in order to replace the [MASK] tokens with correct tokens, a high level of multimodal understanding is required.

With the multimodal text decoder and the associated feature fusion, this approach is able to solve VQA tasks using a simple classification head without changing the overall structure of the model.

Instead of always training the full network, Zhai et al. [95] suggest that for image-text alignment, one freezes the weights for a portion of the network. In the work, the latent space of text and image encoder[7] is also aligned as in the previously named methods, but it is explored whether, and if so what difference it has if one initializes the subnetworks randomly or pre-trained or even pre-trains one of the networks and does not train it further at all during the alignment. It is found that the highest performance is achieved when the vision encoder is pre-trained supervised and all weights are frozen during the alignment, while the text encoder is initialized randomly. This combination is called Locked-image tuning (LiT).

LiT has a zero-shot performance that exceeds that of CLIP [88] and ALIGN [96] in various benchmarks. Even in relatively difficult out-of-distribution (OOD) benchmarks like ObjectNet [97], LiT can show convincing performance. However, a conclusion about VQA capabilities is not made.

Visual Question Answering in Remote Sensing Visual Question Answering for Remote Sensing (RSVQA) was first applied by Lobry et al. [98] as a task to ML. Here, a ResNet-152 [99] pre-trained on ImageNet [74] was used as a visual feature extractor. The last average pooling layer and the fully connected layer

[7]the work uses a dual encoder design

in the classification head were removed. Instead, a fully connected layer was inserted that mapped the 2048-dimensional feature vector to a 1200-dimensional feature vector.

As feature extractor for the text part a RNN was used, which uses the skip-thought architecture [100]. This subnet was pre-trained on the BookCorpus data set [49]. In order to be able to connect the feature vector of this net with that of the ResNet, a fully connected layer was inserted, which mapped the 2400 dimensional vector to a 1200 dimensional one.

Since both nets have the same output dimension, the feature vectors can simply be multiplied together to get a feature combination. Since the two previous fully connected layers are not trained for pre-training but for finetuning, Lobry et al. [98] argue that simple multiplication is sufficient as a late-fusion approach.

To evaluate the capabilities of the network, two data sets are introduced, one consisting of High Resolution (HR) images and one consisting of Low Resolution (LR) images. Both data sets use only Red, Green, Blue (RGB) channels, although the LR data set uses Sentinel-2 data, which is multispectral. The questions created for the HR data set can be divided into 4 categories: Presence ("Is X present?"), Area ("How big is X?"), Counting ("How many X are there?"), and Comparisons ("Are there more X than Y?", "Is X bigger than Y?"). For area, answers are divided into levels (0m^2, 1m^2 - 10m^2, 11m^2 - 100m^2, . . .), counting has answers 0, 1, . . . 89 and for comparisons and presence are yes/no answers.

For the LR the area task is replaced by the question whether a picture is rural or urban. In addition, counting with one number per category is divided into orders of magnitude (0, 1-10, 11-100, . . .), since there were more than 17 000 individual objects in the data set in a single image. However, with 256×256 pixels per image, these can no longer be meaningfully distinguished. The other two categories are the same.

For answering the questions, a MLP is used as a classification head with a hidden layer with 256 dimensions. The input are the fused features (1200 dimensional), the output is one dimension per answer choice.

It was found that the network is very good at distinguishing which task to solve. For example, it rarely answered "Yes" to a counting task. Overall, the network was able to handle the LR data set better than the HR data set, where it visually had the most problems with counting. But also the area determination had a lower accuracy than presence and comparison. The rural/urban task of the LR data set could also be solved relatively well with 90% accuracy.

In a second application with an identical network, Lobry et al. [101] were able to demonstrate again that this network architecture can learn using a BigEarthNet data set [102] for which a data set named RSVQAxBEN containing a total of 15 million question-image-answer triplets were generated from the given labels. However, the data set is a multi-class multi-label data set, so many different label combinations can occur. Here, each combination was considered as a separate class and the 1 000 most frequently occurring classes were evaluated. For questions that could be answered with Yes/No, the network had an accuracy of about

80%, whereas questions about land cover classes were answered correctly with only about 20% accuracy.

Chappuis et al. [103] can detect an improvement in both the LR and HR data sets based on the same architecture by replacing the feature combination method. Here, Multimodal Compact Bilinear Pooling (MCB) [104] and Multimodal Tucker Fusion for Visual Question Answering (MUTAN) [105] are explored as alternatives to feature combination via multiplication. It can be found that both methods have an increase in accuracy over multiplication in all but the rural/urban category. It is believed that this is because this category is relatively rare in the LR data set and therefore the two methods have too little data as both methods are learned. However, MUTAN has fewer parameters than the baseline [98] because this method reduces the 1200 dimensional to 360, thus the fully connected layer contains more parameters. MCB on the other hand has 8000 dimensions as input to the fully connected classification head. Therefore, it is not clear whether the better performance of MCB is due to the increased number of parameters or because it is a better method.

[106] use a different approach to get answers for the RSVQAxBEN data set. Here, a vision network is trained to classify L1, L2, and L3 CORINE Land Cover (CLC) classes like the labels in RSVQAxBEN. The result of the classifier is made into a continuous text by simply converting the corresponding labels into their natural language equivalents and enumerating all positive classes. This enumeration of classes is called the context.

A Distilbert model [50] gets as input the question as well as the context and is supposed to determine from it the answers Yes/No/None as well as all 61 L1/L2/L3 CLC classes as multilabel classification.

In addition to the context obtained from the network, it is tested how well the Distillbert model with a perfect context ("visual oracle") and without context ("visual blind") can answer the questions to show the maximum and minimum performance of this architecture. It can be shown that the trained architecture is better than Lobry et al. [101], but does not reach the maximum performance.

Zheng et al. [107] present a data set derived from the already widely used RS data sets UC-Merced [108], Sydney [109], HRRSD [110], AID [111], and DOTA [112] and combines them and adds automatic and human-made question-answer pairs. This results in a total of over 100 000 image-question-answer tiplets from about 37 000 images. A VGG-16 mesh [113] is used as a feature extractor for the images, and a Gated Recurrent Unit (GRU) is used as a text encoder. Features are combined using an attention block followed by a linear layer to classify the responses. It could be shown that this method works better on the presented data set than single-modality methods, pre-trained text feature extractors like BERT [47] or GoogleNews [114] or other fusion methods like Support Vector Machines (SVMs).

As shown in previous work, different categories of questions are difficult to learn in different ways. Based on this finding, Yuan et al. [115] propose a system in which questions are first learned that have already produced convincing results in other work (e.g., questions about the presence of an object) and more difficult questions (such as questions about the number of objects) are asked at a later stage.

In addition, the data set used requires different scales of objects by combining the HR and LR data sets from [98] and the data set presented in [107]. To handle the different scales, one Cross-Modal Global Attention (CGA) as well as one Cross-Modal Spatial Transformer (CST) module is placed after the CNN in vision feature extractor. These speech-driven image features and question-relevant image regions are recognized by combining the image features with the text features generated by a RNN using attention. Thereby, CGA extracts global attention, while CST extracts regions by attention. Selfpaced Curriculum Learning (SPCL) is used to realize the gradual increase of the difficulty. It could be shown that this method in each category is in part significantly better than the baseline of [98] resp. [107].

Molinier et al. [116] apply visual question answering on change detection with different categories of questions. A data set of about 3000 multi-temporal image pairs with over 122 000 question answer pairs is introduced. The questions are about if a change occurred, if the change was an increase or decrease, what changed, what the smallest and largest classes of change where and what the ratio of changes for different classes was. Answers are given as yes/no, percentage (in 10% steps) or as land cover class.

The architecture used has a multi-temporal CNN encoder with a downstream multi-temporal fusion module, which fuses feature maps of different temporal steps by subtraction. The feature extraction of the question also happens via a RNN, feature fusion of image and text features happens via concatination. The classification head uses several fully connected layers to connect the features and predict the answer as a class.

The architecture used has a multi-temporal CNN encoder with a downstream multi-temporal fusion module, which fuses feature maps of different temporal steps by subtraction. The feature extraction of the question also happens via a RNN, feature fusion of image and text features happens via concatination. The classification head uses several fully connected layers to connect the features and predict the answer as a class. It has been shown that CDVQA is a challenging task, as both small vision encoders like ResNet-18 [99] and parameter-wise larger ones like ResNet-152 or ViT-B16 [73] give only limited good results with an average accuracy below 60% and an overall accuracy below 70% in all cases.

Rahnemoonfar et al. [19] present a data set that includes semantic segmentation and VQA as tasks. The data set consists of 2343 images with a spatial resolution of 15 cm, acquired by Unmanned Aerial Vehicles (UAVs), each with a resoluton of 4000×3000. The images were taken after Hurricane Harvey in August 2017 in Texas and Louisiana, USA and include classes such as (un)flooded houses and streets. Semantic segmentation is enabled by pixel-wise annotation of all images into one of 9 classes. VQA is realized by about 4500 question-image-answer triplets, where each question was created by hand. The questions refer to both local details and global contexts in the images. Questions are divided into 4 categories:

1. *Simple Counting*: Questions about the number of a certain type of object, e.g. houses
2. *Complex Counting*: Similar to *Simple Counting*, but as additional condition the state of the type, e.g. (un)flooded houses
3. *Yes/No*: Questions whether a certain object has a certain condition.
4. *Condition Recognition*: A questions about what condition a certain object type or the whole image have

Answers can be Yes or No for Yes/No questions, flooded, unflooded, or both for condition questions, and numbers from 1 to 41 for counting questions. While Yes/No are relatively balanced as possibilities, there are about 3$\times$ more questions answered with unflooded than those answered with flooded and even less with both. Numerical answers are also strongly unbalanced with 1 to 4 being the most common and larger numbers in particular tending to be exponentially fewer. Note, that 0 does not occur as an answer option. It is also important to note that a question type is always started by the same question word ("How" for counting, "What" for condition, and "Is" for Yes/No).

3 Proposed Lightweight Transformer-based Visual Question Answering Framework for Earth Observation (LiT-4-EOVQA)

Visual Question Answering (VQA) is the task of answering a natural language input of a user with the help of an image. The question refers to the given image. To be able to solve this task, one needs a system, which can convert the different input data into concepts, connect these and evaluate them together. Thus, a system is needed that can 1. handle image data as well as 2. handle natural language and 3. combine knowledge from both domains to obtain a joint answer. The three sub-systems listed are explained and their structure motivated in the following subsections.

To make the system accessible for scenarios where resources are constrained, a lightweight solution in terms of low computational and storage resources is needed. In order to keep this solution flexible and future-proof, the system should be modular and allow the integration of new developments. In the context of this work, the current state-of-the-art for image processing Transformer architecture is used [73]. As image data, satellite-derived data is used for Earth Observation (EO). The system will be referred to as Lightweight Transformer-based Visual Question Answering Framework for Earth Observation (LiT-4-EOVQA) in the following.

3.1 Vision Encoder as Feature Extractor

An elementary part of every solution for visual question answering is the solution of the subproblems of text and image analysis. In order to keep the solutions as modular as possible and to be able to easily integrate future approaches to solving the subproblems into the concept presented here, the problems are treated as independent of each other for the time being.

The extraction of features from an image is therefore realized by training a vision encoder. In principle, this can be implemented by any kind of encoder, but in the context of LiT-4-EOVQA it is specifically defined as a lightweight transformer in the sense of saving computation and memory resources while using a ViT.

To reduce training requirements, the encoder is pre-trained on a data set that does not have to be related to VQA. However, the dimensions of the input data must match that of the final VQA data set used. Otherwise, the embedding produced will not match the embeddings expected by the vision transformer. For EO the inputs are multi- or hyperspectral images, so an encoder trained for RGB data will not work. For this reason it is necessary for LiT-4-EOVQA vision encoder to be trained from scratch.

The encoder can be pre-trained by any type of training such as unsupervised, self-supervised, weakly supervised or supervised. In the context of the training in this paper, the encoder is pre-trained by a multi-label multi-class supervised classification task.

Depending on the selected data set and encoder, the loss functions are freely selectable. For this work the loss function is the weighted Mean Squared Error (MSE).

$$L_{\mathrm{MSE}} = \frac{1}{N_{\mathrm{classes}}} \frac{\sum_{i=0}^{N_{\mathrm{Classes}}} w_i \cdot (y_i - l_i)^2}{\sum_{i=0}^{N_{\mathrm{Classes}}} w_i} \tag{37}$$

Here N_{Classes} is the number of classes used in the pre-training data set, w_i is the weighting of the loss, y is the prediction of the network and l_i the label for class i respectively. This loss emphasizes the prediction error for classes with high weight and de-emphasizes it for classes with lower weight. A typical use case is unbalanced data sets to focus more on under-represented classes by increasing their weights accordingly. For balanced data sets all weights can be set to the same value to have no weighting applied on the loss.

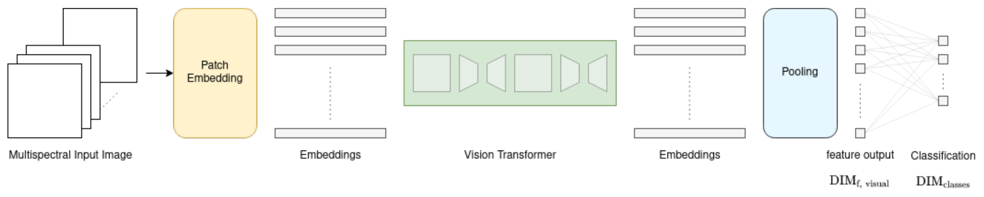

Multispectral Input Image
Patch Embedding
Embeddings
Vision Transformer
Embeddings
Pooling
feature output
$\mathrm{DIM}_{\mathrm{f,\ visual}}$
Classification
$\mathrm{DIM}_{\mathrm{classes}}$

Figure 10: VFormer configured for pre-training. The classification head including its output dimension $\mathrm{DIM}_{\mathrm{classes}}$ is dependent on the data set used for pre-training.

After the completion of the pre-training, the classification head of the VFormer is removed. Thus the final feature map of the encoder is a tensor $x \in \mathbb{R}^{\mathrm{DIM}_{\mathrm{f,\ visual}}}$. Classically, this is the result of pooling layers on the final embeddings after they have passed through all transformer layers as seen in fig. 10. Some architectures don't use polling as a final layer. Otherwise the last filter layer is used. The Transformer layer keeps the trained knowledge, which is used in the subsequent task as initialization for further task-specific fine-tuning. As long as the multispectral input for the fine-tuning task has the same dimensionality as the input of the pre-training, the knowledge in the patch embedding can still be used.

However, in some cases it is advantageous if the pre-training takes place in the same domain as the fine-tuning [117, 118]. So it is not necessarily sufficient to pre-train for a twelve-dimensional final task on twelve-dimensional images, but it would make more sense to pre-train and fine-tune within the same domain, e.g. satellite data. It is also preferable that both data sets have similar features such as spatial resolution and spectral bands in order to be able to apply the knowledge from the pre-training directly. Alternatively, a non-domain-specific pre-training followed by a domain-specific pre-training can be combined to prepare the encoder for the following task of VQA.

3.2 Text Encoder as Feature Extractor

Similar to the vision encoder for visual features, the text encoder must be pre-trained in order to reliably extract information from the input given in text form. Unlike the vision encoder, however, the text input for domain-specific texts is not of a different dimensionality than for non-domain-specific texts. Therefore, for the text encoder, any system can be used that allows the encoder to convert the text input into an embedding with high information content.

The encoder can be trained independently from the actual task. It does not have to be able to handle the domain, Remote Sensing, nor the type of input, questions, since the text encoder is only concerned with the ability to develop a general understanding of the text. As long as the understanding can be given as output in the form of embedding, information can be extracted from a question and implemented in a later processing step.

This domain-unspecific pre-training has the great advantage that it is not necessary to collect data for the application in the territory of Earth Observation. Instead, models pre-trained on generall-purpose data sets with extensive knowledge about the structure of a language can be used.

In addition, this has the advantage that the application of this system can change the language by replacing the text encoder. Although it is necessary to fine-tune again after the exchange, since embeddings are not guaranteed to be transferable from one language to another, this avoids the computationally and time intensive language model pre-training step. This is particularly relevant for the encoders used in LiT-4-EOVQA, as they use KD and therefore a LLM must first be trained before training the small model, which is particularly resource and time intensive.

The encoder can be implemented in different ways. For LiT-4-EOVQA the encoder is implemented by a lightweight text transformer. This encodes information in pre-training for classification tasks by converting the input into tokens and placing a special token `[CLS]` before the first token. Only this is considered in the classification header as shown in fig. 11.

Flattening the final encoding of the `[CLS]` token produces a vector of dimension $\mathbb{R}^{\text{DIM}_{\text{f, text}}}$. A simple linear layer classifies the feature output. This classification head is removed similar to the one of the vision encoder for the following fine-tuning step. Thus, the output of the text encoder is a $\text{DIM}_{\text{f, text}}$-dimensional vector, which has encoded a large part of natural language relevant features from the input by its pre-training.

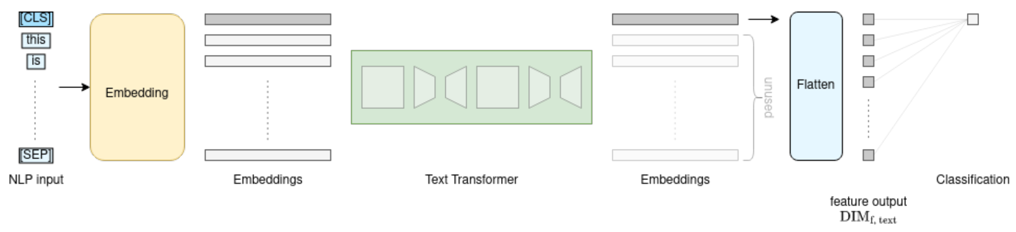


Figure 11: QFormer configured for pre-training. The classification head is dependent on the task used for pre-training. Here, the configuration is displayed for a binary classification task like NSP or SOP.

3.3 Late Feature Fusion

Late feature fusion is applied so that pre-trained encoders can be used for the different modalities. The fusion is dependent on the outputs of the individual networks. This ensures that the same principle can be used regardless of the encoder.

For the fusion, the heads of the individual encoders are removed. This leaves the pre-trained stem, which in the following acts as a feature extractor for its respective modality. A linear layer is inserted to map the outputs of the encoders, which have the dimensions $\mathrm{DIM}_{\mathrm{f,\,text}}$ and $\mathrm{DIM}_{\mathrm{f,\,visual}}$, to a constant dimension $\mathrm{DIM}_{\mathrm{in}}$ for each of the two modalities. These two linear layers are the only encoder-dependent stages of the late fusion part of LiT-4-EOVQA. All other dimensions are freely adjustable independent of the encoders.

Another advantage of these mappings is that the sometimes very high-dimensional feature outputs of the encoders can be reduced to a smaller dimensionality. This reduces the memory and computational requirements of the entire network.

The used Feature Fusion Method (FFM) can be configured arbitrarily as long as it meets the condition that it can map an input vector with $2 \times \mathrm{DIM}_{\mathrm{in}}$ dimensions to an output vector with $\mathrm{DIM}_{\mathrm{in}}$ dimensions. Half of the input vector consists of image encoder features and half of text encoder features. Possibilities of the fusion can be for example point-wise multiplication as used by Lobry et al. [98], element-wise addition of the two feature vectors or also a learned mapping. For the mapping the features would be concatenated and put into a linear layer with corresponding in- and output dimensions.

The last part of a LiT-4-EOVQA network is a classification head. In this thesis, it is implemented in the form of an MLP with a hidden layer to keep the structural complexity and computational requirements low. The hidden layer is specified according to Lobry et al. [98] with the dimension $\mathrm{DIM}_{\mathrm{hidden}} = \mathrm{DIM}_{\mathrm{in}}/2$. The final layer is to be set depending on the data set. Each answer choice gets an output neuron. Thus a multi-label multi-class classification is possible, but only limited in the context of the training data. Free text answers, especially zero-shot answers are not possible with this system. However, the classification head can

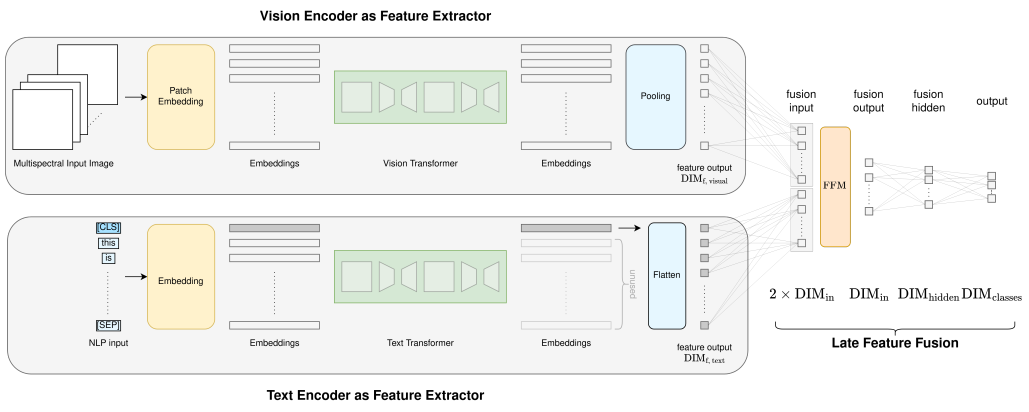


Figure 12: LiT-4-EOVQA network architecture for fine tuning. Feature output dimensions $\text{DIM}_{\text{f, visual}}$ and $\text{DIM}_{\text{f, text}}$ are dependent on the vision and text encoders used while DIM_{in}, $\text{DIM}_{\text{hidden}}$ and FFM are configuration parameters of the feature fusion. $\text{DIM}_{\text{classes}}$ is dependent on the data set used for application.

be adapted according to the task without changing the overall functionality of the system.

To avoid co-adaptation of individual units in the linear layers and the associated overfitting, a dropout layer [119] is placed on each linear layer. Furthermore, each linear layer is activated by a non-linearity. This strategy is also used for the case when a linear layer is used as FFM.

Since the special nonlinearity is independent of the rest of the network, the effect of function selection is discussed in this work. For the purpose of simplification, dropout and activation layer are not drawn in fig. 12.

The network is trained using weighted MSE

$$L_{\text{MSE}} = \frac{1}{N_{\text{classes}}} \frac{\sum_{i=0}^{N_{\text{Classes}}} w_i \cdot (y_i - l_i)^2}{\sum_{i=0}^{N_{\text{Classes}}} w_i} \tag{38}$$

as loss function with the weights w_i dependent on the answer distribution in the training data.

Thanks to its modular design, the system presented has the advantage that new technical developments can be incorporated at any time. This flexibility also makes it possible to interconnect systems of any complexity, as long as the interfaces adhere to the given specifications. The linear layers introduced act as adapters to make these connections as simple as possible.

In addition, the standardized interface of the modules to each other creates the possibility to try out different combinations of networks in an efficient way. This allows the different networks tested in this work to be quickly interconnected and different combinations to be tried out, while the structure of the overall network remains the same, while it also makes LiT-4-EOVQA particularly suitable for iterative development of VQA networks on the basis of existing feature extractors. The benefits of the presented framework are used in this thesis to test and compare the networks presented in section 2 in different combinations with different classification heads.

4 Data Set Description

A common approach for training of Deep Neural Networks (DNNs) like CNNs or Vision Transformers is pre-training on large annotated RGB data archives like ImageNet [74] or proprietary data sets like the Google internal JFT-300M data set used by Kuznetsova et al. [73]. However, DL approaches in RS often use multispectral or hyperspectral images for improved performance on DL networks [10, 120, 121].

As shown by Neumann et al. [117] and Risojević and Stojnić [118] pre-training on domain-specific data sets can improve the final performance of DNNs. This chapter presents the BigEarthNet benchmark archive used for pre-training as well as two VQA data sets for finetuning which have been created from the archive.

4.1 BigEarthNet Classification Data Set

Sumbul et al. [102] introduced a data set based on 125 tiles of Sentinel-2 image data, each spanning a surface area of 10km $\times$ 10km. Sentinel-2 images are composed of 13 bands in 10m, 20m and 60m spatial resolution. These include Visible Light Spectrum (VIS) (380 nm – 700 nm), Near Infrared Light Spectrum (NIR) (700 nm – 1100nm) and Short-Wave Infrared Light Spectrum (SWIR) (1100 nm – 3000 nm) bands. 4 Bands (B02 – blue, B03 – green, B04 – red and B08 – HR-NIR) have a spatial resolution of 10m and are designed to conform to the conditions of LCC. Additional 6 Bands (B05, B06, B07, B08a, B11 and B12) were arranged for the monitoring of vegetation in 20m spatial resolution. These bands are located at the Vegetation red-edge and SWIR for Cloud, Snow and Ice discrimination. Bands B01, B09 and B10 are designed to cover atmospheric parameters like aerosols, water-vapor and cirrus clouds and have a 60m spatial resolution [43].

The tiles are divided into 590 326 patches, where each patch covers a surface area of 1200m $\times$ 1200m, corresponding to 120 $\times$ 120 pixels, 60 $\times$ 60 pixels and 20 $\times$ 20 pixels for 10m, 20m and 60m bands respectively. Each patch is annotated with one or multiple labels of the hierarchical CLC database, which contains 43 classes in its most detailed Level 3. For the construction of the BigEarthNet archive, band 10 was not included as it contains no surface information.

A second revision [122] expandes the data set by adding a Sentinel-1 patch of 10m spatial resolution (120 $\times$ 120 pixels) for each Sentinel-2 patch in the set. The Sentinel-1 and Sentinel-2 patches map the same geological location with close acquisition times, thereby being processed as a pair of image patches.

Sentinel-1 images are acquired using a Synthetic Aperture Radar (SAR) in different modes depending on the observation scenario [123]. Most data acquisition over Europe is done using the Interferometric Wide Swath (IW) mode in dual polarization (VV and VH), thereby creating two SAR-images. This data is in-cooperated into the multi-modal BigEarthNet (BigEarthNet-MM) archive.

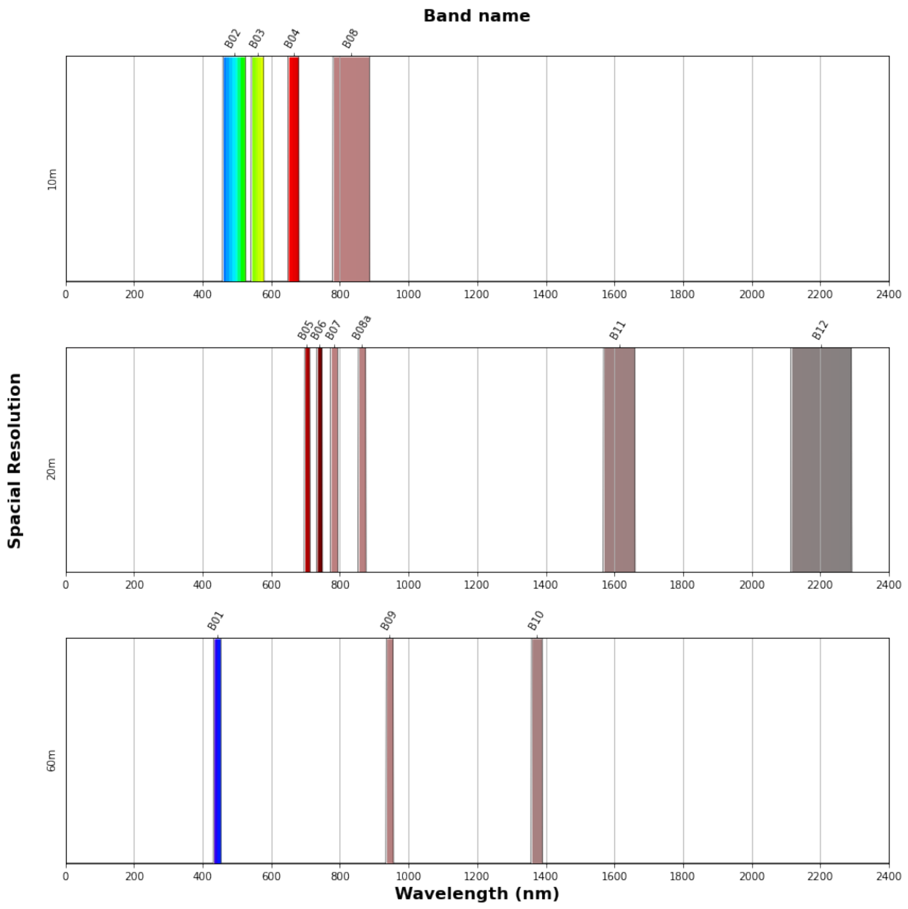


Figure 13: Spectral Bands of Sentinel-2 Bands for different Spatial Resolutions. [43, fig. 5]

An alternative nomenclature for the labels annotates each with up to 19 labels based on the CLC database. In this nomenclature, a total of 57 images had no label anymore. Additionally, 9 280 (partially) covered by clouds or shadow and another 61 707 images are covered by seasonal snow. Therefore, it is recommended not to use these images. This results in a total of 519 284 image pairs in the recommended set of images.

This data set was used for pre-training of all Vision Transformers. However, only the 10 meter and 20 meter bands of the Sentinel-2 images and both bands of the Sentinel-1 images where used. Therefore, a total of 12 Bands was used for pre-training.

All bands were converted from 12 bit unsigned to 32 bit float for interpolation to a size of 120 $\times$ 120 pixels with bicubic interpolation and value-inverted in the case of Sentinel-1 images. This was done, as Sentinel-1 image values are recorded as decibel and therefore negative. As the interpolation may cause values

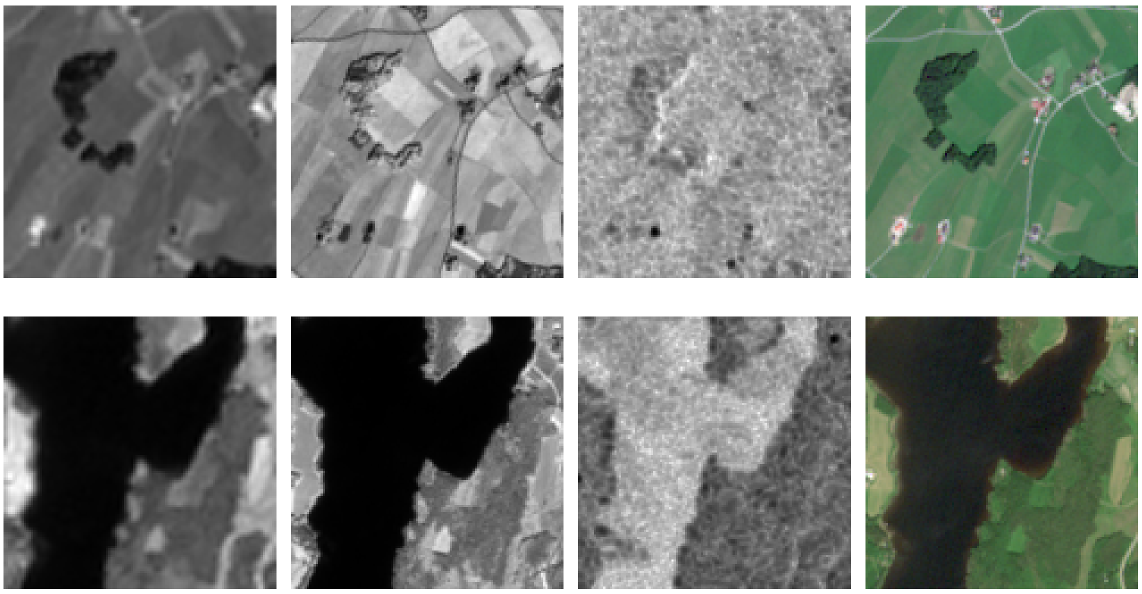

(a) Sentinel-2 Band 5 (20m, Vegetation Red-edge) (b) Sentinel-2 Band 8 (10m, HR-NIR) (c) Sentinel-1 VV Polarisation (d) Sentinel-2 True Color Image (B04, B03 & B02)

Figure 14: Example images from the BigEarthNet-MM data set. Selected channels/channel combinations only.

that are smaller than zero or greater than 1 due to overshoot, image values where scaled to a range from zero to one.

4.2 Proposed BigEarthNet-MM-VQA Data Set

Previously published data sets for RSVQA include images covering the visible spectrum [19, 98, 101, 107, 116]. In addition, all images are in the RGB-channel format commonly used in everyday life. Although the advantages provided by Multi-Spectral Imagery (MSI) and Hyper-Spectral Imagery (HSI) that a more detailed distinction of different materials despite having the same “color” in the spectrum visible to the human eye, especially in the Remote Sensing (RS) community have long been known [102, 124–128], to my knowledge there is currently no multi- or hyperspectral data set for VQA. Therefore, in the context of this thesis the BigEarthNet-MM-VQA data set based on the images of the BigEarthNet-MM data set. The data set is extended by a catalog of questions and answers for each image to form the BigEarthNet-MM-VQA data set. In order to diversify the linguistic variety of the questions, a new question generation concept was created, which can present a question with identical content in many different ways. This increased linguistic difficulty is another distinction from other previously published data sets.

All available images from the BigEarthNet-MM data set were used as a basis. Although questions were generated for all images, as in the original data set, it is recommended not to use images that are completely covered with snow or obscured by a cloud or cloud shadow. Also, no images that do not get a label after conversion from the 43-label version to the 19-label version are included in the data set.

This results in a total of 519 284 images in the data set. By keeping the original split, 269 695 images are used for training, 123 723 for validation and 125 866 for testing. As for the pre-training data set, the 10m and 20m bands as well as the two Sentinel-1 bands, i.e. 12 bands in total, were used for this data set. The same interpolation and value range adjustment was used as for the pre-training data set. This concept also introduces new types of questions that are not yet present in any VQA data set for RS.

Question construction The generation of the questions belonging to the images runs in a multi-stage process. In the process, random decisions are made at several points. To make the question generation repeatable, the random number generators are seeded.

There are five types of questions, with one question type generated in a single stage, three question types generated in two stages, and one question type generated in more than two stages. The question types are:

- *Presence*: Question about what classes there are
- *Single*: Question about which classes are left excluding an existing class
- *Double*: Ask which classes are left excluding two existing classes
- *Multiple*: Ask which classes are left excluding three or more existing classes
- *Choice*: yes/no question, possibly additional explanatory answers expected

In the first stage, one of these types is chosen at random. *Choice* is the most complex case. Slightly more than 33% of all questions belong to this type. In 1/3 of all cases this type is used immediately, in 2/3 of the cases the number of labels on the image currently to be questioned has an influence on the question type. If the image has only one label, only questions of type *Single* and *Presence* will make sense. For questions of the category *Double* and *Multiple*, there must be 2 or more or 3 or more ground truth labels, respectively. If 2 or more labels belong to the image, the *Choice* category is again a choice option to make the question catalog more challenging.

The categories *Single*, *Double* and *Multiple* exclude one, two or more than two labels from the list of available labels and then ask which labels are still available. Answers for these questions consist of all existing but not excluded labels of the image. If there is no label left, there is an additional answer option “No Answer”.

The category *Presence* simply asks which land cover classes are visible on the image. It is similar to the three categories above with the difference that no class is excluded. The answers of this category are identical to those of a normal classification task on the given image.

Questions of the type *Choice* are questions that can be answered with yes or no. There are two sub-types of *Choice* questions: questions that ask for a certain combination of existing classes and questions that ask for remaining classes similar to the types *Single*, *Double* and *Multiple*, but still include a counting task. Which of these subtypes is chosen depends on how many classes are excluded from the question. Unlike the *Single*, *Double* and *Multiple* types, however, not only classes that are actually on the image are excluded here, but classes from the complete class catalog are selected.

If no class is excluded, which is the case in 4/11 of all questions, a specific combination of classes is asked for. For this, one, two or three classes are chosen from the complete catalog. These classes are then connected with either "and" or "or" to ask for a combination of classes. For "and" questions, the answer is yes if and only if all of them are present, while for "or" questions, the answer is yes if at least one of the enumerated classes is present. If not all respectively none of the mentioned classes belong to the shown image, then the answer to the corresponding question is no.

For questions of the *Choice* type in which classes are excluded, one to four classes are excluded with decreasing probability. The probability for one class is 3/11, the probability for two is 2/11 and the probability for three and four is 1/11. 50% of the cases are then asked whether there is at least one other class in the image. If not, the answer is no. If there is at least one additional class, yes is expected along with a list of all additional classes present. Alternatively, in the other half of the cases, it is asked if there is more than one additional class. The difference here is that this question should be answered no if there is only one additional class. If there are two or more additional classes, yes is also expected with an enumeration of the classes.

Since the additional yes/no answer is expected depending on how many additional classes there are and the yes/no distinction is not simply dependent on the number of classes, but also on the exact subtype of the question (at-least-one subtype and more-than-one subtype), these questions are more complex than the *Single*, *Double* and *Multiple* types. Furthermore, since all classes can be excluded, the variety of questions for the *Choice* type is greater than for the other 4 types. The full process of question generation is shown in fig. 15.

Questions are constructed using a building block system of sentence parts to be grammatically correct. For each part of the sentence there are different possibilities in order to be able to ask questions that are as linguistically diverse as possible. For example, for each question type there are several ways to introduce the question, i.e. to choose the first few words.

Moreover, an introduction is not exclusive for a single question type. This is to avoid that it can be recognized already by the first word of the question which question type it is, as it is the case in the data set presented by Zheng et al. [107]. 4 questions were generated per image. This results in 1 078 780 questions in the training set, 494 892 questions in the validation set and 503 464 questions in the test set, i.e. a total of over two million image-question-answer triplets.

Questions/Answers distributions Individual sets were created for each of the splits train, validation and test. Due to the random generation, the ratios of the questions are not identical, but differ by less than one percent from the theoretical percentages. The distribution for the sub-types of the questions of type *Choice* for the training set can be seen in fig. 16b. Again, the ratios in the splits are not exact, but also vary only by parts of percentages. Distributions for validation and test split are provided in the Appendix (fig. 34, fig. 36).

For the training set, 533 956 unique questions were generated, with the most common questions being verbatim-similar questions of type *Presence*. The same is true for the validation set with 289 000 unique questions and the test set with 293 421 unique questions. The absolute frequencies of the 200 most common questions for the training set are shown in fig. 17a. The drop around the 95-most frequent question is the change from questions of type *presence* to other types. All questions to the left of the drop are of type *Presence* while all to the right are also of other types. This is due to the limited content-related diversity of questions of type *Presence*, since these questions all ask the same thing but are phrased differently. However, this also shows the linguistic diversity of the proposed data set.

The most frequent answer combinations in all three sets are “No”, “No answer”, and “Marine waters” with 178 951 times “No”, 126 184 times “No answer”, and 44 529 “Marine waters” in the training set, 81 668 times “No”, 58 322 times “No Answer” and 21 015 times “Marine waters” in the validation set, and 83 420 times “No”, 59 799 times “No Answer”, and 21 252 times “Marine waters” in the test set. The distribution of the 50 most frequent response combinations for the training set are shown in fig. 17b. Overall the training, validation and test set contain 12 872, 8 035 and 8 344 unique answer combinations respectively.

Therefore, it is necessary not to consider the answer combinations as one class per combination, otherwise the data set would become highly unbalanced. Instead, the classes should be considered individually. In fig. 17c the distribution of the individual classes in the training set is shown. This distribution is also highly unbalanced, which should be kept in mind while using the data set. Details on distributions in validation and test set are provided in the Appendix (fig. 35, fig. 37).

Four question-answer pairs have been generated for each image, with the type of question being re-determined at each generation. Examples of image-question-answer triplets are shown in fig. 18.

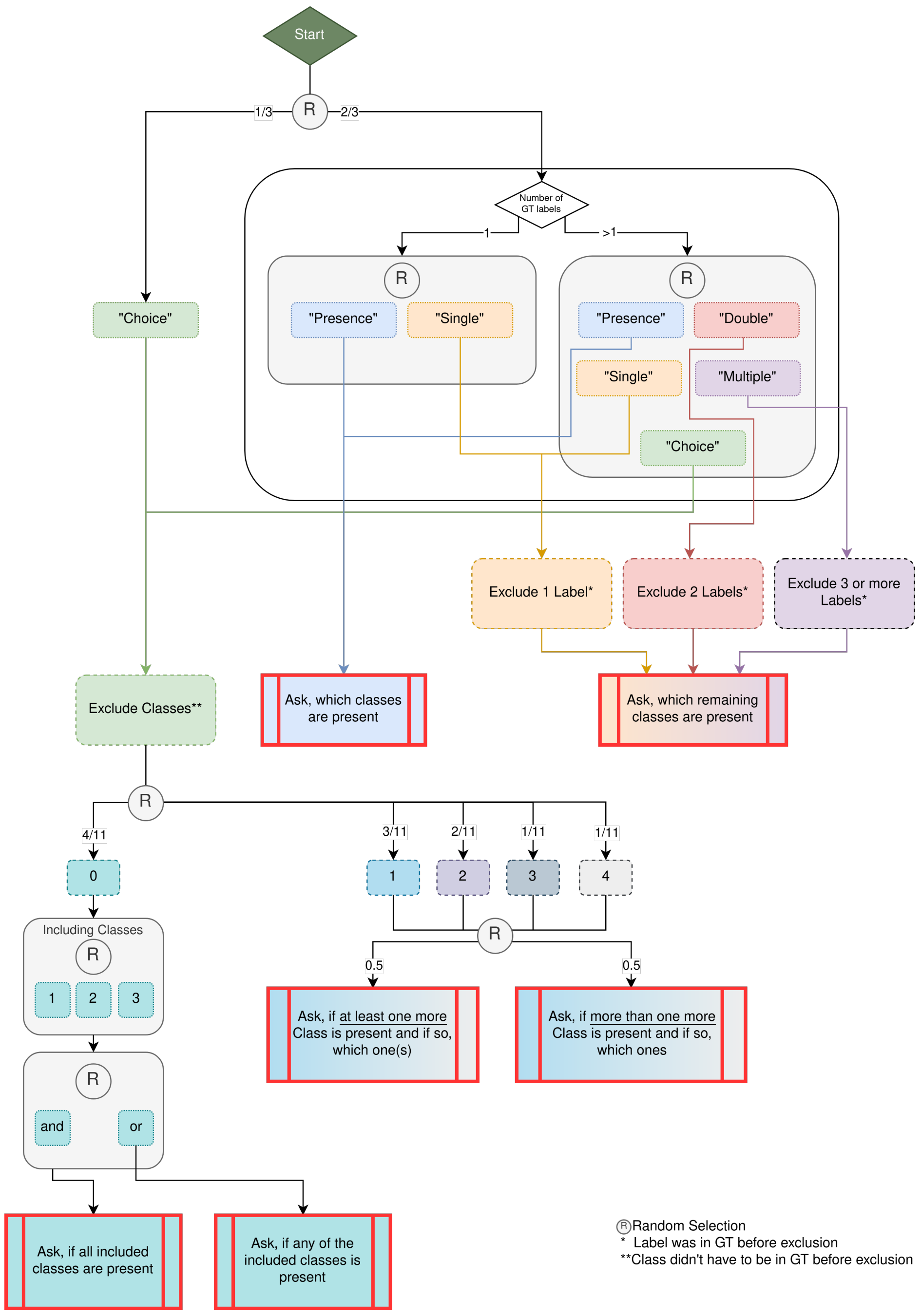
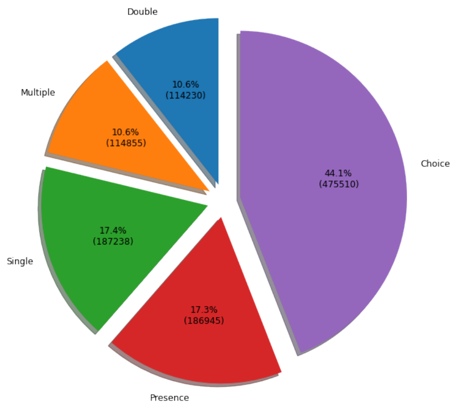
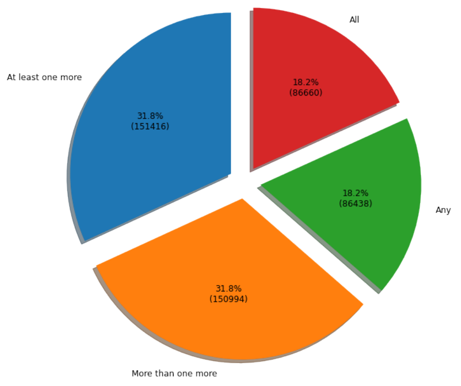

Start
1/3
R
2/3
Number of GT labels
1
>1
R
"Presence"
"Single"
R
"Presence"
"Double"
"Single"
"Multiple"
"Choice"
"Choice"
Exclude 1 Label*
Exclude 2 Labels*
Exclude 3 or more Labels*
Exclude Classes**
Ask, which classes are present
Ask, which remaining classes are present
R
4/11
3/11
2/11
1/11
1/11
0
1
2
3
4
Including Classes
R
1
2
3
R
0.5
0.5
Ask, if at least one more Class is present and if so, which one(s)
Ask, if more than one more Class is present and if so, which ones
R
and
or
Ask, if all included classes are present
Ask, if any of the included classes is present
Ⓡ Random Selection
* Label was in GT before exclusion
**Class didn't have to be in GT before exclusion

Figure 15: Flow diagram of Question Construction – In the first stage, the general type of question is assigned. Some question types can be refined with a second stage, in which, based on the type, the question is detailed by including or excluding a set of (possibly existing) classes and/or specifying the type of the class more precisely. Options within gray boxes are selected at random with equal probability.

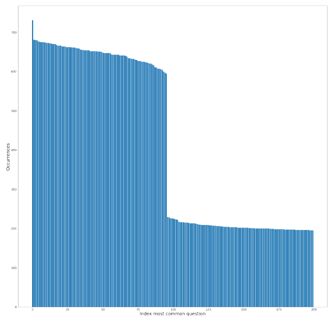


(a) Question type distribution for the main types *Choice*, *Presence*, *Singe*, *Double* and *Multiple* for the training set.

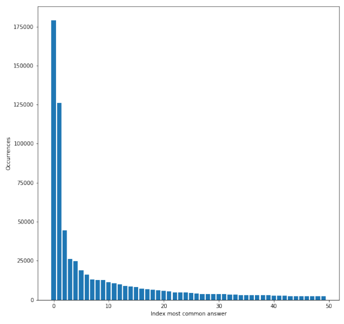


(b) Question sub-type distribution for the main type *Choice* for the training set.

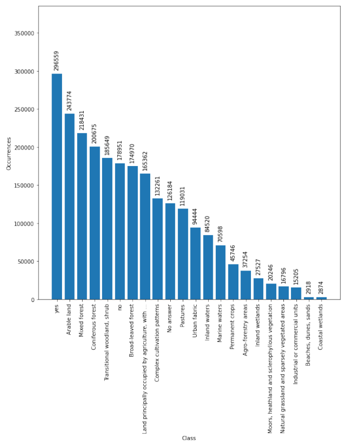
Figure 16: Question type distributions for the training set.



(a) Absolute frequencies of most common questions for the training set. Questions to the left of the drop around index 95 are about *Presence*. Questions to the right of the drop are about other types.

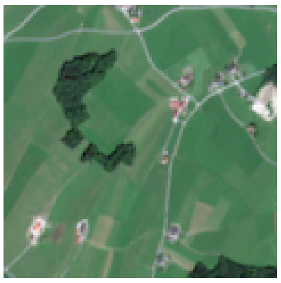
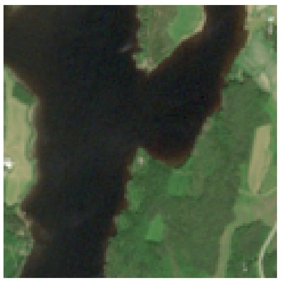


(b) Absolute frequencies of most common answer combinations for the training set. Most common combinations are "No", "No Answer", and "Marine waters" in all sets.



(c) Absolute frequencies of classes in the training set. Class name "Land principally occupied by agriculture, with significant areas of natural vegetation" shortened to "Land principally occupied by agriculture, with . . . ".

Figure 17: Absolute frequencies of *n* most common Questions and Answers in the training set.

	“Besides Pastures and Land principally [. . .], what kind of class are visible here”	“Mixed forest”
	“Barring Mixed forest, which land cover class are visible in this scene?”	“Pastures”, “Land principally [. . .]”
	“Except Transitional woodland, shrub and Complex cultivation patterns, is there a Land cover class visible in this image”	“Yes”, “Arable land”, “Mixed forest”, “Inland waters”
	“Besides Land principally [. . .] and Inland waters, are there LCs visible here”	“Yes”, “Arable land”, “Mixed forest”

(a) Sentinel-2 True Color Image (B04, B03 & B02)

Figure 18: Example VQA triplets from the BigEarthNet-MM-VQA data set. Only true color image shown.
Images: `S2A_MSIL2A_20170613T101031_51_78` (top) and `S2A_MSIL2A_20170701T093031_19_58` (bottom).
Classname “Land principally occupied by agriculture, with significant areas of natural vegetation” shortened to “Land principally [. . .]”.

5 Experimental Results

5.1 Design of Experiments

This chapter describes the training procedures of the pre-training of the vision feature extractors and the results of the pre-training. In addition, the training procedures for LiT-4-EOVQA-networks and the trained combinations as well as the evaluation metrics are presented.

5.1.1 Vision Transformer Pre-training

In order to train fewer parameters, the feature extractor networks are pre-trained separately. Since the Text Feature Extractors have already been trained, there is no need to train them again. However, since the Vision Feature Extractors have been changed in structure to accept the multi-spectral images of the used RS data set, they must be trained from scratch.

For this training, the weights are initialized according to their respective authors (e.g. CSWin uses a truncated normal distribution to initialize some parameter groups) and then pre-trained. This section presents results of the pre-training of the in section 2.2.4 presented architectures in the visual component of the proposed pipeline. All architectures are trained using the BigEarthNet-MM data set introduced by Sumbul et al. [122] with modifications as presented in section 4.1.

The networks are trained for 100 epochs with validation every epoch. AdamW [129] was used as optimizer with Linear Warmup Cosine Annealing Learning Rate and weight decay of 0.01. Maximum learning rate and Warmup are set on a network to network basis on the basis of empirical results. The batch size was set to 256 for all pre-training of all networks. Training was seeded with seed `4242` using `seed_everything()` by `pytorch_lightning` [130]. The training state with the smallest average validation loss was used as final network.

The networks are evaluated using the loss, precision, recall, accuracy, the F1 score and the Receiver Operating Characteristic (ROC)-Area Under Curve (AUC)-Score [131]. If utilized, True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) use a threshold of 0.5 while True Positive Rate (TPR) and False Positive Rate (FPR) are calculated at different thresholds to calculate the ROC curve. The ROC-AUC-Score is the integral of the ROC curve for a FPR from 0 to 1. All evaluation metrics are averaged over all classes.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{39}$$

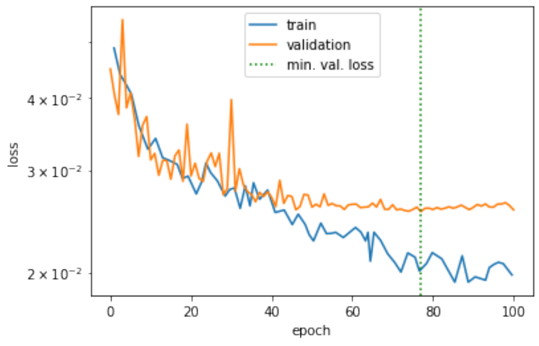
$$\text{TPR} = \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{40}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{41}$$

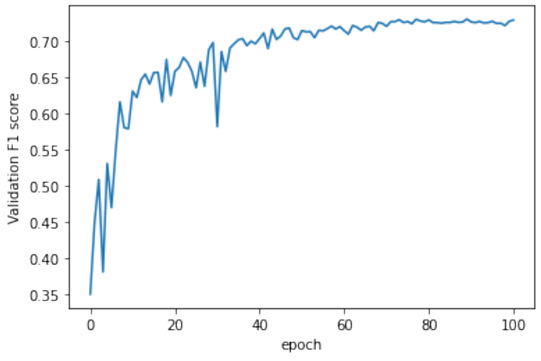
$$\text{Precision} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{42}$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{43}$$

MobileViT A learning rate of 0.001 was used with Warmup for five epochs. As displayed in fig. 19a the training loss decreases over the full training while the validation loss reaches a steady state after about 60 epochs with the minimum at 77 epochs. F1 Score increases over the full training with only small improvements after the point of smallest validation loss (see fig. 19b). On the test set, the network has an F1 score of 0.7316, which is the highest among the trained Vision Feature Extractors. It also reaches the highest scores on recall and accuracy as shown in table 1.



(a) Training and validation loss for MobileViT pre-training. Dotted green line marks the state of network with the lowest validation loss.

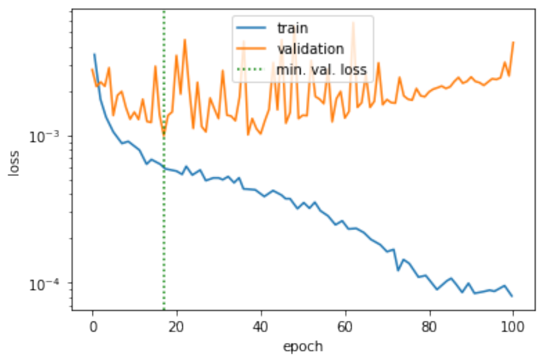


(b) F1 Score on the validation set for MobileViT pre-training.

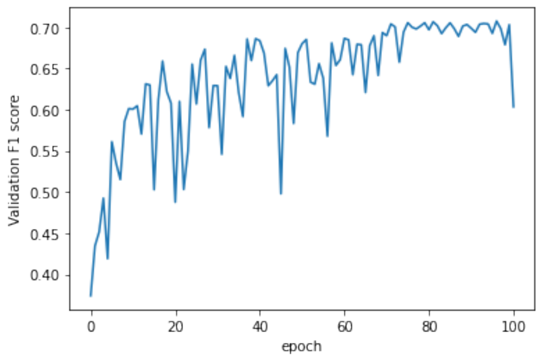
Figure 19: Pre-training statistics for MobileViT network.

MobileFormer The learning rate 0.0008 was used with Warmup for two epochs. The network reaches its minimal validation loss after 17 epochs while training loss decreases and F1 score increases for the rest of the training. An F1 score of 0.6653 is reached on the test set. The networks has the best performance for loss, the ROC-AUC-Score and precision among the trained vision feature extractors (see table 1).

CSWin Learning rate was set to 0.00025 with Warmup for five epochs. The lowest validation loss is reached after 43 epochs, after which both the training loss continues to decrease and the validation F1 score continues to increase



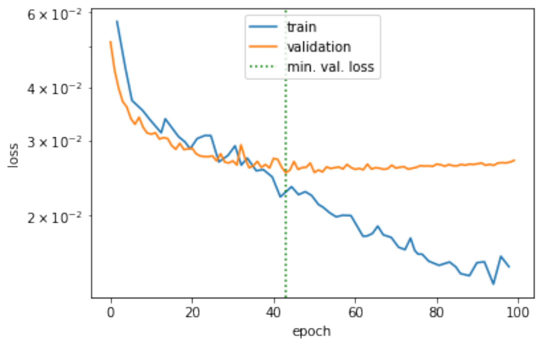
(a) Training and validation loss for MobileFormer pre-training. Dotted green line marks the state of network with the lowest validation loss.



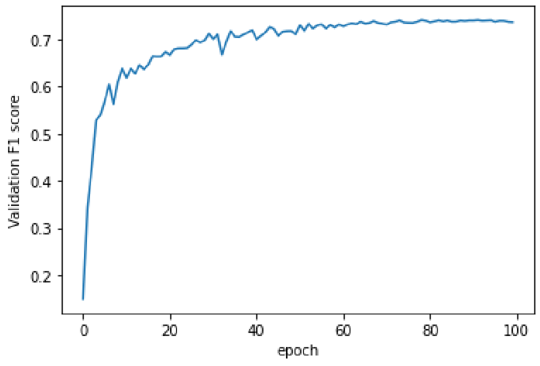
(b) F1 Score on the validation set for MobileFormer pre-training.

Figure 20: Pre-training statistics for MobileFormer network.

slightly as seen in fig. 21. The network achieves a test F1 score of 0.7293. The network has the second best scores in all evaluation categories, thereby never being the best but also never worse then any but one (see table 1).



(a) Training and validation loss for CSWin pre-training. Dotted green line marks the state of network with the lowest validation loss.

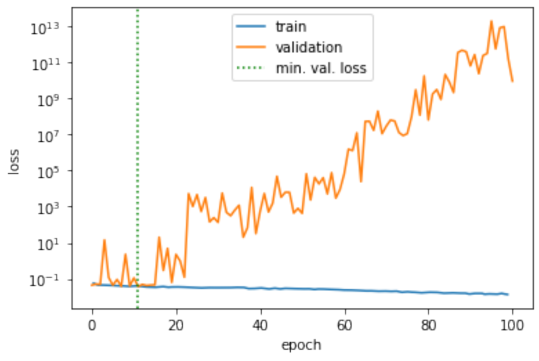


(b) F1 Score on the validation set for CSWin pre-training.

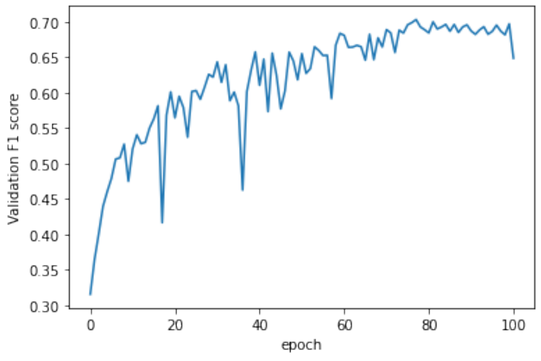
Figure 21: Pre-training statistics for CSWin network.

ConvMixer A learning rate of 0.0008 was utilized with Warmup for five epochs. The network reaches the lowest validation loss after 11 epochs, after which the validation loss increases exponentially. However, the network seems to continue learning, as both the training loss decreases (see fig. 22a) and the F1 score increases (see fig. 22b). In accordance with the other networks, the state with the lowest validation loss was nevertheless finally evaluated. The network achieves an F1 score of 0.5507 on the test set.

The network has the lowest performance in all evaluation metrics among the trained vision feature extractors as visible in table 1. For uniformity, the training loss is shown in fig. 19a together with the validation loss. Since the details of the training loss are lost, it is shown again individually in the appendix in fig. 33.



(a) Training and validation loss for ConvMixer pre-training. Dotted green line marks the state of network with the lowest validation loss.



(b) F1 Score on the validation set for ConvMixer pre-training.

Figure 22: Pre-training statistics for ConvMixer network.

Table 1: Results on the test set for pre-training of the VFormer networks.

Network	Loss	ROC-AUC-Score	Precision	Recall	Accuracy	F1 Score
MobileViT	0.02613	0.9446	0.7794	**0.6949**	**0.9371**	**0.7316**
MobileFormer	**0.00099**	**0.9593**	**0.8448**	0.5694	0.9329	0.6653
CSWin	0.02585	0.9485	0.7862	0.6869	0.9365	0.7293
ConvMixer	0.04025	0.9117	0.6851	0.4871	0.9117	0.5507

5.1.2 LiT-4-EOVQA Transformer Training

Different configurations of LiT-4-EOVQA networks are trained. The configurations each differ by one aspect, while all other aspects are kept the same. All configurations are trained with the AdamW optimizer [129] with a weight decay of 0.01. The learning rate is set depending on the configuration. All networks are trained for 20 epochs and the latest state is evaluated with the test set. The loss, Mean Average Precision (mAP)-Micro and -Macro the F1 score and the ROC-AUC score [131].

mAP-Micro and -Macro differ in that mAP-Macro calculates the mAP for each class individually and then averages it, whereas mAP-Micro does not consider class-specific differences. To calculate the mAP, the area under the precision-recall curve is considered, where the curve is formed by forming different thresholds.

As different configurations, networks with different Visual Feature Extraction Sub-Networks (VFormers), Question Feature Extraction Sub-Networks (QFormers), Activation Functions in the Late Fusion module and Feature Combination Methods (FCMs) including dimensions of the Linear Layer in the Classification Head ($\mathrm{DIM}_{\mathrm{hidden}}$) as well as Feature Fusion Methods (FFMs) are considered. A full overview of which configurations are tested including the specifications of all network parts is shown in table 2.

Table 2: EOVQA model configurations being evaluated. Duplicates grayed out but displayed in their respective block for completeness.

Special Characteristic	Network Name		Activation Function	FCM			Configuration Name
	VFormer	QFormer		$\mathrm{DIM}_{\mathrm{in}}$	FFM	$\mathrm{DIM}_{\mathrm{hidden}}$	
VFormer	MobileViT	BERTTiny	Tanh	256	Mult	128	MVBT-T256M128
	MobileFormer	BERTTiny	Tanh	256	Mult	128	MFBT-T256M128
	CSWin	BERTTiny	Tanh	256	Mult	128	CSBT-T256M128
	ConvMixer	BERTTiny	Tanh	256	Mult	128	CMBT-T256M128
QFormer	MobileViT	BERTTiny	Tanh	256	Mult	128	MVBT-T256M128
	MobileViT	TinyBERT	Tanh	256	Mult	128	MVTB-T256M128
	MobileViT	Distilbert	Tanh	256	Mult	128	MVDB-T256M128
	MobileViT	Mobilebert	Tanh	256	Mult	128	MVMB-T256M128
	MobileViT	Albert	Tanh	256	Mult	128	MVAB-T256M128
Activation Function	MobileViT	BERTTiny	Tanh	256	Mult	128	MVBT-T256M128
	MobileViT	BERTTiny	ReLU	256	Mult	128	MVBT-R256M128
	MobileViT	BERTTiny	GELU	256	Mult	128	MVBT-G256M128
	MobileViT	BERTTiny	Sigmoid	256	Mult	128	MVBT-S256M128
	MobileViT	BERTTiny	ELU	256	Mult	128	MVBT-E256M128
FFM	MobileViT	BERTTiny	Tanh	256	Mult	128	MVBT-T256M128
	MobileViT	BERTTiny	Tanh	256	Add	128	MVBT-T256A128
	MobileViT	BERTTiny	Tanh	256	CatNet	128	MVBT-T256C128
Fusion Layer Dimensions	MobileViT	BERTTiny	Tanh	256	CatNet	128	MVBT-T256C128
	MobileViT	BERTTiny	Tanh	512	CatNet	256	MVBT-T512C256
	MobileViT	BERTTiny	Tanh	128	CatNet	64	MVBT-T128C64

5.2 Results on Visual Feature Extraction Sub-Network Variations

In these configurations, the vision feature extractor is different for each configuration. For the configurations with MobileViT (MVBT-T256M128) and ConvMixer (CMBT-T256M128) the learning rate is set to 0.001. The configurations with MobileFormer (MFBT-T256M128) and CSWin (CSBT-T256M128) networks have a learning rate of 0.0005.

As seen in fig. 23, all configurations reach a converged state in validation loss after about 5 epochs, while the training loss continues to decrease over the entire training. The final loss on the test set is between 0.0179 and 0.0203 with MVBT-T256M128 configuration having the lowest loss and CMBT-T256M128 having the highest loss. MVBT-T256M128 also achieves the best result for the metrics mAP-micro and ROC-AUC-score, with MVBT-T256M128 achieving the highest mAP-micro and CSBT-T256M128 the highest F1 score. Unlike in the pre-training, the ConvMixer configuration CMBT-T256M128 is not always the worst in all metrics. Although the result in mAP-Macro with approx. 0.72 is significantly worse than in the other configurations with approx. 0.76-0.77, the network achieves a higher ROC-AUC score than the MobileFormer (MFBT-T256M128) and CSWin (CSBT-T256M128) configurations.

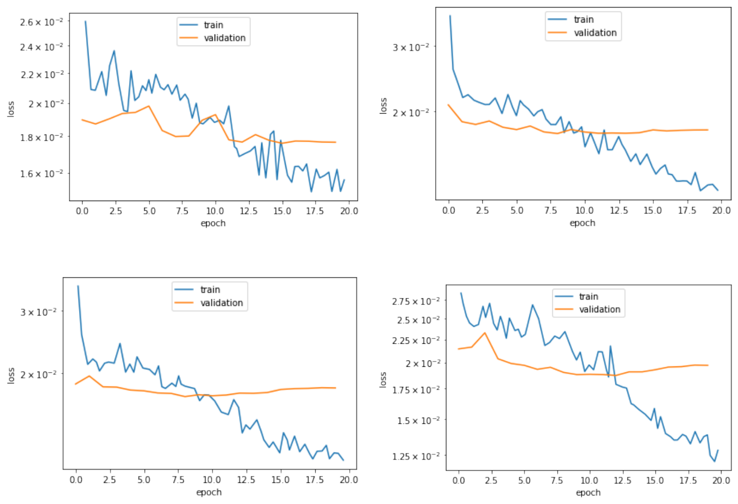


Figure 23: Training and validation loss for MobileViT, MobileFormer, CSWin and ConvMixer Variations (in reading order). All other configuration parts of the LiT-4-EOVQA network are kept the same.

As seen in fig. 24, the validation F1 score improves steadily during training for all configurations, while other metrics reach their highest validation performance already between the 5th and 12th epoch. It is particularly noticeable that MVBT-T256M128 achieves the best results in mAP-Micro, mAP-Macro and the ROC-AUC score towards the end of the training, while all other networks already lose in performance. It is also noticeable that the ConvMixer configuration CMBT-T256M128 starts with significantly worse results in the earlier epochs in every metric, but catches up in all except mAP-Macro, and in some cases even delivers better results in the meantime. The validation performance of CMBT-T256M128 for the metric mAP-Micro is even higher than for MVBT-T256M128 in epoch 10, although MVBT-T256M128 delivers the best mAP-Micro results on the test set at the end of the training.

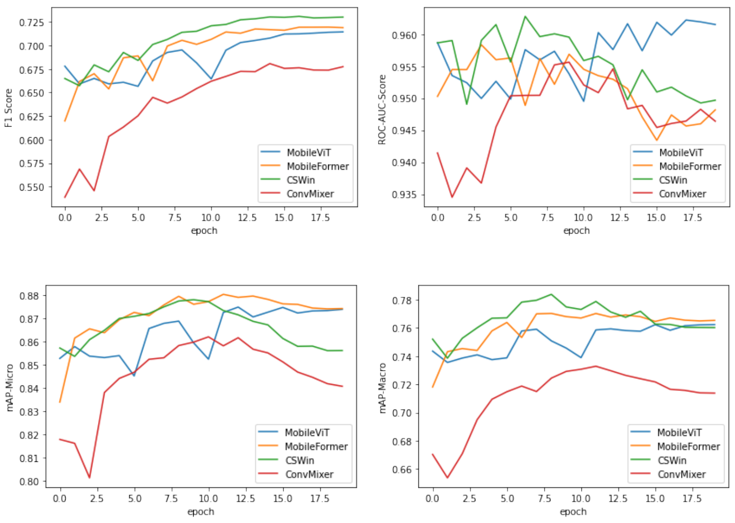


Figure 24: Comparison of F1 Score, ROC-AUC-Score, mAP-Micro and mAP-Macro for all VFormer Variations (in reading order). All other configuration parts of the LiT-4-EOVQA network are kept the same.

Table 3: Results on the test set training of LiT-4-EOVQA networks with different Vision Feature extractors.

Configuration Name	Loss	mAP-Micro	mAP-Macro	ROC-AUC-Score	F1 Score
MVBT-T256M128	**0.0179**	**0.8754**	0.7686	**0.9599**	0.7180
MFBT-T256M128	0.0182	0.8740	**0.7710**	0.9476	0.7228
CSBT-T256M128	0.0185	0.8562	0.7613	0.9460	**0.7344**
CMBT-T256M128	0.0203	0.8402	0.7183	0.9479	0.6802

5.3 Results on Question Feature Extraction Sub-Network Variations

In the configurations compared here, only the text feature extractors are replaced, while the other configuration parts remain the same. The BERT$_{\text{Tiny}}$ (MVBT-T256M128), TinyBERT (MVTB-T256M128) and Albert (MVAB-T256M128) configurations are trained with a learning rate of 0.001, the Mobilebert (MVMB-T256M128) configuration is trained with 0.0003 and the Distilbert (MVDB-T256-M128) configuration with a learning rate of 0.0001.

As shown in fig. 25, all configurations reach a converged state after 12 epochs at the latest. The MVMB-T256M128 configuration already reached the state after 3 epochs, while the other networks only reached the state after later training steps. A difference between this configuration and the others is, that this configuration also has a converged state in the training loss, while the other networks continue to train. It can also be seen that the loss of the MVMB-T256M128 and the

MVAB-T256M128 configurations is significantly higher than for the other three configurations.

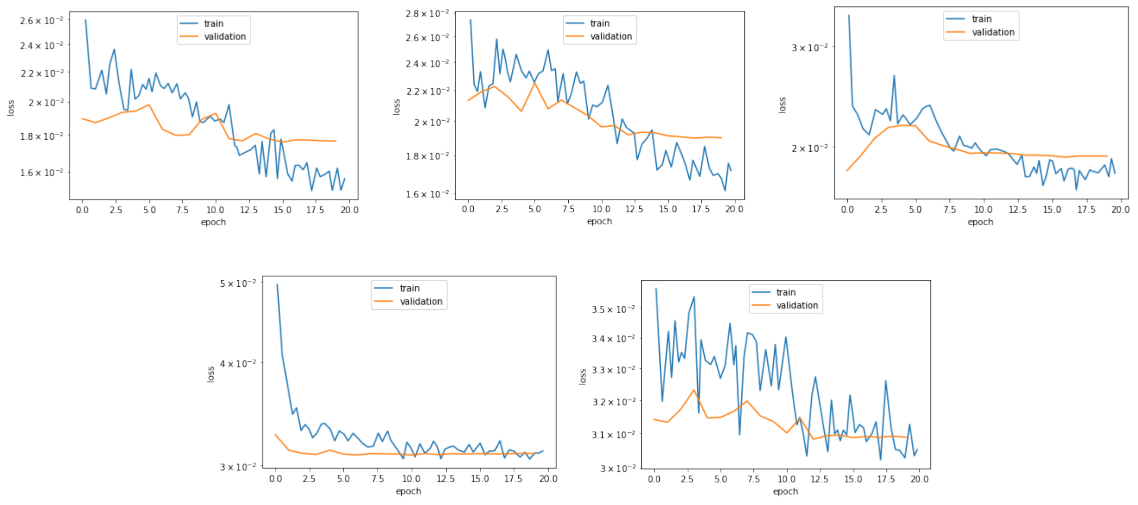


Figure 25: Training and validation loss for BERT$_{\text{Tiny}}$, TinyBERT, DistilBERT, MobileBERT and Albert Variations (in reading order). All other configuration parts of the LiT-4-EOVQA network are kept the same.

This can also be seen in the results on the test set and the metrics during the validation phases. While the MVBT-T256M128, MVTB-T256M128 and MVDB-T256M128 configurations are in the same order of magnitude, the MVMB-T256M128 and MVAB-T256M128 configurations are significantly worse in the context of the metrics. The MVBT-T256M128 is the best in all metrics, while the MVTB-T256M128 and MVDB-T256M128 deliver partly identical results to the third decimal place. The results of configuration MVBT-T256M128 is with 0.8754 up to 3% (mAP-Micro) and with 0.9599 only 0.5% (ROC-AUC-Score) better than the second best result performing configuration MVTB-T256M128 with 0.8452 and 0.9549 respectively.

The results of the MVMB-T256M128 and MVAB-T256M128 configurations are considerably worse than the next best network. For example, both configurations score less than half as well in the F1 score with 0.3322 and 0.3281 for MVMB-T256M128 and MVAB-T256M128 respectively, while the other three networks have scores between 0.6930 and 0.7180. The biggest absolute difference is in the mAP-Micro metric where both networks score almost 37% less than the third best network configuration MVDB-T256M128 as seen in table 4.

This difference is not only present during the test phases, but also during the full training as shown in fig. 26. The two networks always have the difference in each metric after the first, last, and in every epoch in between, which is also evident during the test phase. During training, sometimes one network and sometimes the other is better at intermediate states in terms of the metric. Similarly, with the MVTB-T256M128 and MVDB-T256M128 configurations, sometimes one network is better and sometimes the other. while the MVBT-T256M128 configuration achieves the highest value at almost every point during training. Only after the first epoch this configuration is not highest in the mAP metrics. From the second epoch onwards, no other network achieves the same values as the MVBT-T256M128 configuration for this metric, which is ultimately also reflected in the final test result, where there are 3% difference between this and the second best performing network for both metrics.

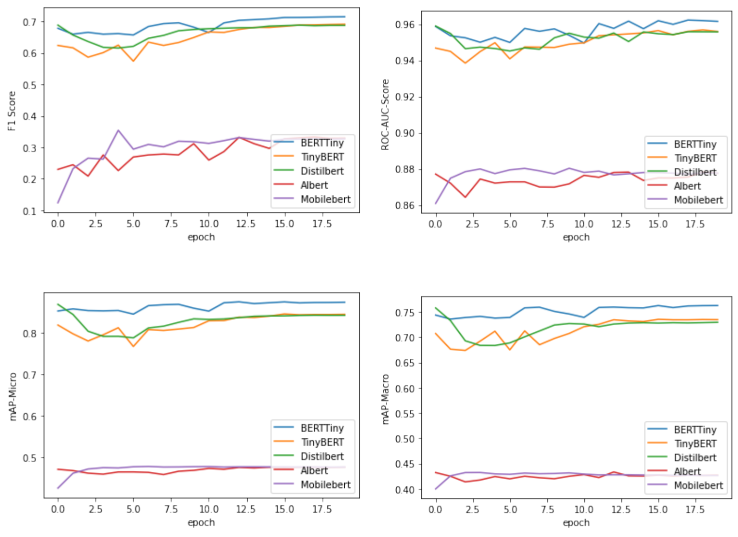


Figure 26: Comparison of F1 Score, ROC-AUC-Score, mAP-Micro and mAP-Macro for all QFormer Variations (in reading order). All other configuration parts of the LiT-4-EOVQA network are kept the same.

Table 4: Results on the test set training of LiT-4-EOVQA networks with different Text Feature extractors.

Configuration Name	Loss	mAP-Micro	mAP-Macro	ROC-AUC-Score	F1 Score
MVBT-T256M128	**0.0179**	**0.8754**	**0.7686**	**0.9599**	**0.7180**
MVTB-T256M128	0.0194	0.8452	0.7396	0.9549	0.6930
MVDB-T256M128	0.0196	0.8420	0.7315	0.9546	0.6933
MVMB-T256M128	0.0315	0.4754	0.4300	0.8774	0.3322
MVAB-T256M128	0.0313	0.4764	0.4324	0.8775	0.3281

5.4 Results on Activation Function Variations

In the configurations tested in this section, various activation functions were tested in the Late Feature Fusion module of the LiT-4-EOVQA models. As Li et al. [132] and others have shown, the choice of activation function can have an impact on the capabilities of a network. However, since different subnetworks used in this thesis use different nonlinearities, e.g. MobileViT uses Swish [133] and MobileFormer uses Dy-ReLu [80], different possibilities are explored in this chapter. All used activation functions do not change the parameter count of the respective network.

Activation functions were only exchanged in the linear layers of this module, i.e., between the feature extractors and the fusion input, between fusion output and the fusion hidden layer, and between the hidden layer and the output layer. Any Activation Fuctions in the feature extractors were not exchanged. The learning rate for the Tanh-using network MVBT-T256M128 was set to 0.001, while the other networks with nonlinearities ReLU (MVBT-R256M128), GELU (MVBT-G256M128), Exponential Linear Unit (ELU) (MVBT-E256M128) and Sigmoid (MVBT-S256M128) had a learning rate of 0.0003.

As seen in fig. 27, the qualitative progression of all activation functions is roughly the same. It is only noticeable that the validation loss for the MVBT-S256M128 network is lower than the training loss at every point in time, while for each of the other configurations there is a point in time where the training loss becomes lower than the validation loss. In addition, the quantitative analysis shows that the loss of this network-configuration over the entire training is about twice as high as for the other networks.

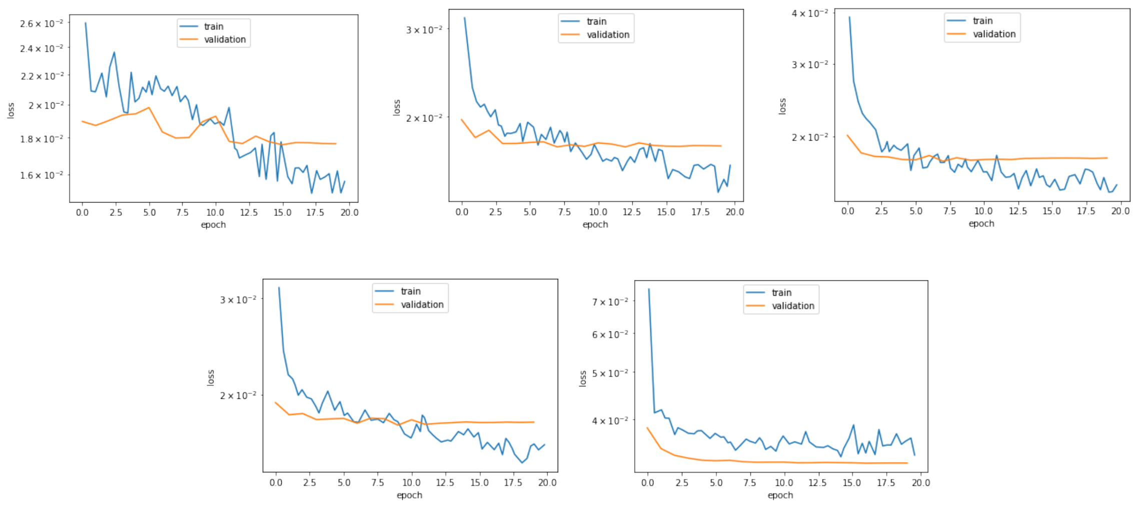


Figure 27: Training and validation loss for Tanh, ReLU, GELU, ELU and Sigmoid Variations (in reading order). All other configuration parts of the LiT-4-EOVQA network are kept the same.

A similar behavior can also be seen in the validation phases for all other metrics, as shown in fig. 28. While the Linear Units configurations and the Tanh network provide competitive results, the Sigmoid network-configuration performs worse by a large margin. Except for the F1 score, this worse score is constant for the entire training. Only the F1 score improves with longer training, being zero or close to zero for the first 8 epochs.

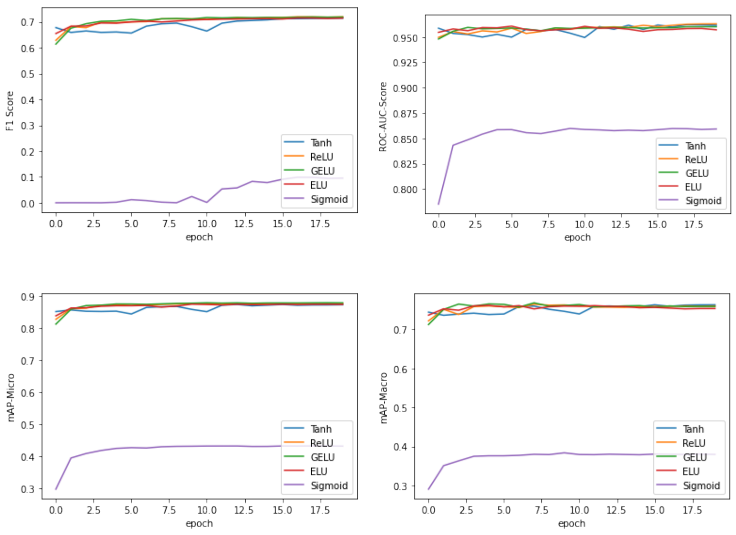


Figure 28: Comparison of F1 Score, ROC-AUC-Score, mAP-Micro and mAP-Macro for all Activation Function Variations (in reading order). All other configuration parts of the LiT-4-EOVQA network are kept the same.

The large differences between the MVBT-S256M128 configuration and the other configurations also carry over into the test phase (see table 5). The test loss of approx. 0.033 is almost twice as high as that of all other network-configurations with approx. 0.018. The greatest difference is also apparent here in the F1 score, in which the MVBT-S256M128 configuration with 0.0957 achieves less than a seventh of the score of the second worst configuration MVBT-E256M128 with 0.7176.

The other four configurations are often less than one percent apart in the evaluated metrics in the test phase. The biggest difference is in the metric mAP-Macro, where the best configuration MVBT-T256M128 with 0.7686 is about 1.3% better than the worst of the four (MVBT-E256M128 with 0.7558). In all other metrics, the best and worst network of the four configurations differ by less than one percent. Furthermore, not always the same configuration is the best, but depending on the metric, one or the other network is. Only the MVBT-E256M128 configuration is not the best in any metric, but always the worst, excluding the previously examined Sigmoid configuration.

Table 5: Results on the test set training of LiT-4-EOVQA networks with different Activation Functions.

Configuration Name	Loss	mAP-Micro	mAP-Macro	ROC-AUC-Score	F1 Score
MVBT-T256M128	**0.0179**	0.8754	**0.7686**	0.9599	0.7180
MVBT-R256M128	**0.0179**	0.8774	0.7561	**0.9628**	0.7203
MVBT-G256M128	0.0181	**0.8807**	0.7600	0.9589	**0.7226**
MVBT-E256M128	0.0182	0.8753	0.7558	0.9565	0.7176
MVBT-S256M128	0.0331	0.4309	0.3816	0.8575	0.0957

5.5 Results on Feature Fusion Method Variations

The networks considered in this section use different methods to combine features from the VFormer and the QFormer. These experiments will investigate whether simple element-wise operations are sufficient to link multidimensional features of different domains, or whether more complex linking operations should be used to obtain promising results. The element-wise multiplication (MVBT-T256M128) has trained with a learning rate of 0.001, while the other two methods element-wise addition (MVBT-T256A128) as well as CatNet (MVBT-T256C128) each have a learning rate of 0.0003. CatNet means that the two feature vectors are con*CAT*enated and then a *NET*work consisting of a linear layer is used to bring them to the correct dimensionality and to combine the features in a linear way.

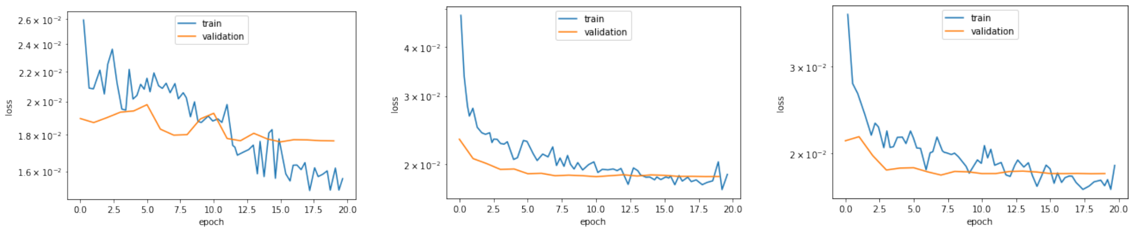


Figure 29: Training and validation loss for Multiplication, Addition and CatNet Feature Fusion Method Variations (in reading order). All other configuration parts of the LiT-4-EOVQA network are kept the same.

As seen in fig. 29, the qualitative trajectories of the trainings of all three networks are similar, with only the loss being different especially in the first few epochs, converging as the training progresses. This behavior can also be observed during the validation of the metrics F1 Score, mAP-Micro and mAP-Macro. As seen in fig. 30, the validation results for the MVBT-T256A128 and MVBT-T256C128 configurations start more than ten percent below that of MVBT-T256-M128 in some cases. However, all configurations have similar performance towards the end of the 20-epoch training, with element-wise multiplication having

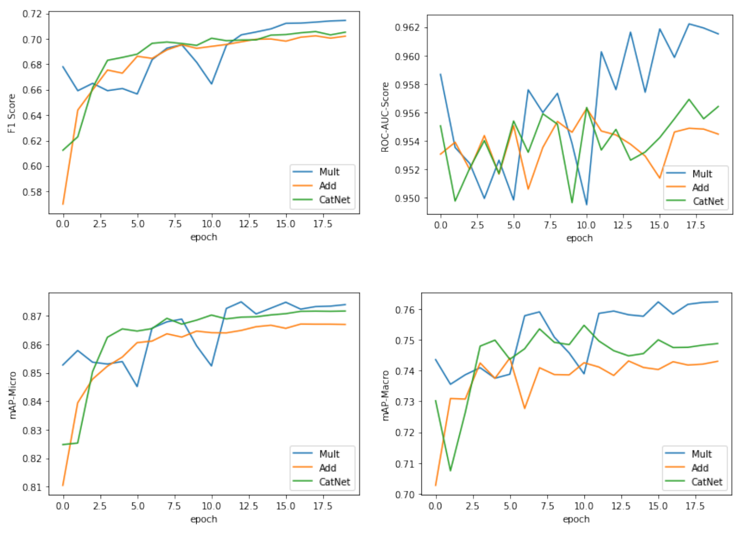


Figure 30: Comparison of F1 Score, ROC-AUC-Score, mAP-Micro and mAP-Macro for all Feature Fusion Method Variations (in reading order). All other configuration parts of the LiT-4-EOVQA network are kept the same.

higher results than CatNet and CatNet higher than element-wise addition for all metrics.

MVBT-T256M128 is not the best network at all times during training. In the validation phases of epochs 5 and 10, the performance in all metrics is lower than for the other two configurations. In the subsequent epochs, however, the performance is again partly competitive and partly better than that of the MVBT-T256A128 and MVBT-T256C128 configurations. Overall, it can be seen that the MVBT-T256M128 configuration had the greatest fluctuations from epoch to epoch in the metrics when comparing the qualitative validation performance curves with the other two configurations.

This order also carries over into the test phase. The results plotted in table 6 show that the MVBT-T256M128 network achieves the best result in all metrics. However, the difference between this network and the worst, in all cases MVBT-T256A128, is less than two percent in all metrics, in some cases significantly less. The largest difference is in the metric mAP-Macro with about 1.55% and the footnotesizeest in the ROC-AUC-score with less than 0.6%.

5.6 Results on Fusion Layer Dimensions Variations

In the configurations compared here, the Feature Fusion Method (FFM) is always fixed to CatNet and the dimensions of the Late Feature Fusion module $\mathrm{DIM}_{\mathrm{in}}$ and $\mathrm{DIM}_{\mathrm{hidden}}$ are changed. Due to the changes of $\mathrm{DIM}_{\mathrm{in}}$, the output dimensions of

Table 6: Results on the test set training of LiT-4-EOVQA networks with different Feature Fusion Methods.

Configuration Name	Loss	mAP-Micro	mAP-Macro	ROC-AUC-Score	F1 Score
MVBT-T256M128	**0.0179**	**0.8754**	**0.7686**	**0.9599**	**0.7180**
MVBT-T256A128	0.0190	0.8670	0.7421	0.9541	0.7075
MVBT-T256C128	0.0186	0.8717	0.7492	0.9543	0.7085

the Feature Extractor Networks $\text{DIM}_{\text{r, visual}}$ and $\text{DIM}_{\text{r, text}}$ are also adjusted accordingly. The variations in the classification head will investigate whether a change in the computational capacity of the network after the fusion of the modalities has an impact on the answering capabilities of the network. Since the changes in dimensions are also accompanied by a change in the number of parameters, a trade-off analysis must be made.

The configurations are referenced as $\text{DIM}_{\text{in}}/\text{DIM}_{\text{hidden}}$, so 256/128 denotes the configuration $\text{DIM}_{\text{in}} = 256$ and $\text{DIM}_{\text{hidden}} = 128$. The learning rate for all three configurations 256/128 (MVBT-T256C128), 512/256 (MVBT-T512C256) and 128/64 (MVBT-T128C64) is set to 0.0003.

Both the qualitative course of the training and validation loss of all three networks is roughly the same. They differ only in the quantitative comparison, where as in fig. 31 the MVBT-T128C64 configuration has the highest training loss while that of MVBT-T256C128 and MVBT-T512C256 are very similar, with the course of MVBT-T512C256 being somewhat flatter. Since the validation loss of all networks has the same constant order of magnitude from about the third epoch, it is noticeable that the MVBT-T256C128 network has a higher validation loss than training loss after about 8 epochs, while for MVBT-T512C256 this point is reached only after about 15 epochs. The MVBT-T128C64 configuration has a higher training loss than validation loss during the whole training, but for all networks no convergence of the training loss can be seen yet, while the validation loss is already fully converged after 10 epochs.

Looking at the metrics in fig. 32, it is noticeable that after about 10 epochs, the ranking of the networks as a function of performance in the metrics F1 score, mAP-Micro and mAP-Macro is clear. Here, the larger network achieves a higher score than the smaller network in each comparison. Overall, this relationship is valid for almost the entire course of the training, whereby only for a maximum of a few epochs are individual relationships reversed before the larger network achieves better results again. Only in the ROC-AUC-score is this relationship only valid in the last two epochs, whereby the differences between the networks can only be found in the third decimal place.

This ranking in the validation metrics can also be transferred to the test phase. As can be seen in table 7, the MVBT-T512C256 configuration is leading in all metrics and only in the ROC-AUC-score is the MVBT-T128C64 configuration equal, with the worst configuration in this comparison MVBT-T256C128 with 0.9543 only 0.1% below the result of the other two networks with 0.9553 each. The

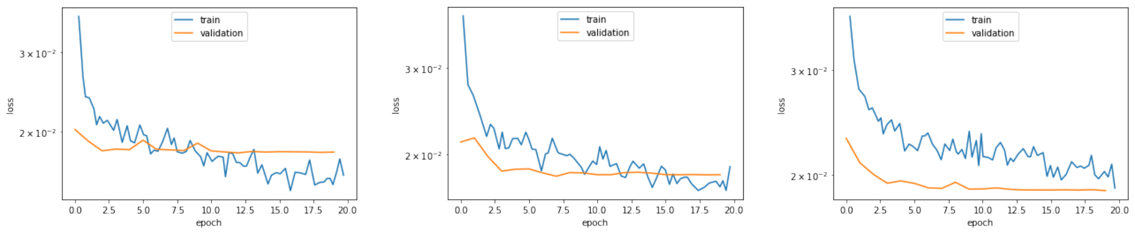


Figure 31: Training and validation loss for 512/256, 256/128 and 128/64 Fusion Layer Dimension Variations (in reading order). All other configuration parts of the LiT-4-EOVQA network are kept the same.

biggest difference between the networks can be seen in the F1 score, where the largest and best network MVBT-T512C256 with 0.7189 is almost one percent ahead of the second largest and second best network MVBT-T256C128 with 0.7085. The footnotesizeest network MVBT-T128C64 is more than four and three percent behind the best and second best networks with 0.6741, respectively.

Table 7: Results on the test set training of LiT-4-EOVQA networks with different Fusion Layer Dimensions.

Configuration Name	Loss	mAP-Micro	mAP-Macro	ROC-AUC-Score	F1 Score
MVBT-T256C128	0.0186	0.8717	0.7492	0.9543	0.7085
MVBT-T512C256	**0.0184**	**0.8774**	**0.7543**	**0.9553**	**0.7169**
MVBT-T128C64	0.0192	0.8616	0.7362	**0.9553**	0.6741

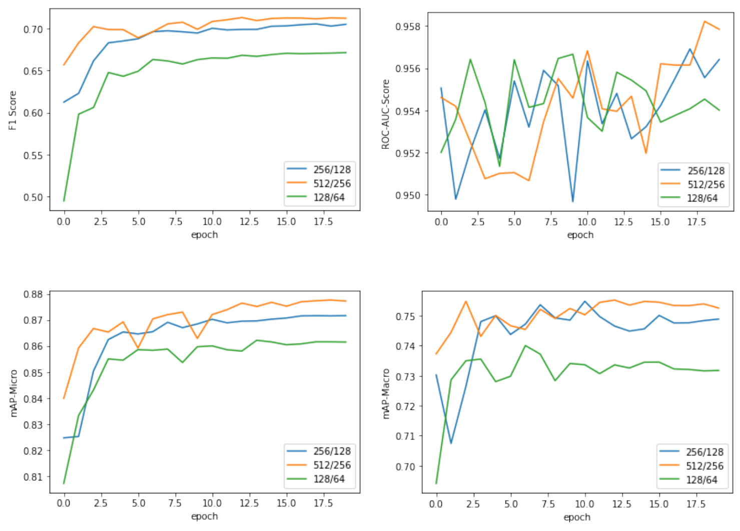


Figure 32: Comparison of F1 Score, ROC-AUC-Score, mAP-Micro and mAP-Macro for all Fusion Layer Dimension Variations (in reading order). All other configuration parts of the LiT-4-EOVQA network are kept the same.

6 Conclusion and Discussion

In this thesis, a Lightweight Transformer-based Visual Question Answering Framework for Earth Observation (LiT-4-EOVQA) was proposed. To demonstrate the functionality of the framework, the data set multi-modal BigEarthNet Visual Question Answering (BigEarthNet-MM-VQA) was created. The different aspects of the framework were evaluated by examining the individual components. This involved testing individual parts of the network like Visual Feature Extraction Sub-Networks (VFormers), Question Feature Extraction Sub-Networks (QFormers), Feature Combination Methods (FCMs) split into Feature Fusion Methods (FFMs) and fusion layer dimensions respectively, as well as the effect of using different activation functions in the late fusion approach. The flexibility of LiT-4-EOVQA allowed the connection of different networks.

During the investigation, it was noticed that pre-training of the VFormer has an influence on the final result after finetuning, but that finetuning can also deliver competitive results with sub-networks that are not optimally pre-trained. The situation is different for the choice of QFormer. Here, the sub-network has a significant influence on the result, with some networks being significantly worse or not trainable at all than others. A more in-depth exploration of why some QFormers are not suitable for training with LiT-4-EOVQA or what adaptations are needed could be explored in a future work.

While investigating different activation functions in the late feature fusion module of LiT-4-EOVQA networks, it was found that both section-wise linear activation functions and the Tanh function are suitable. However, the Sigmoid function was not trainable to a comparable performance. One possible explanation for this behavior is that the Sigmoid function is the only one of the functions used here that is neither unbounded to positive nor has a 0 mean. Whether the networks behave in the same way for other activation functions with these properties and whether this is in fact the cause of the observed training progressions could be investigated in a future work.

Point-wise addition, point-wise multiplication, and a linear layer were investigated to examine different feature combination methods. The best method in the evaluated metrics is the multiplication, followed by the linear layer. The fact that the linear layer is better than the point-wise addition can be explained, since the linear layer can learn a point-wise addition and must be thus in no case worse. However, a single linear layer is not able to learn the multiplication due to the use of the lipschitz continuous activation functions [134]. In order to be able to approximate this, at least one hidden layer is necessary, whereby here the number of the hidden neurons have a large influence [135].

Whether a CatNet modification with hidden layer can provide a better fusion than the methods explored here could still be explored. In addition, further training with and without L2 regularization, and with and without weight clipping, could investigate whether Lipschitz continuity is the origin of the behavior observed in these networks.

The last investigated variable of LiT-4-EOVQA based networks is the dimensionality of the layer in the late feature fusion module. It has been found that a higher dimension leads to better results. Due to the large data set introduced in this work, this behavior may be explained by the scalability of DL-networks, as is the case of large language models, since these networks learn to generalize better with more parameters when training on a larger data set. However, this theory works against the aspiration of using lightweight transformers made in this work. Therefore, this approach will not be pursued further in the context of this work, but could be explored in more detail in a future work and alternatives to simple scaling could be searched for. The configurability of the approach presented in this thesis, especially the MVBT-T256M128 configuration, which has been identified as the best performing of the ones tested, especially considering the low parameter count of the configuration, as well as the data set designed for VQA for RS, can be used as a basis for future research in this regard.

References

[1] Statista.com. *YouTube: Hours of video uploaded every minute 2020*. Statista. URL: https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute/ (visited on 05/05/2022).

[2] DLR - Earth Observation Center. *500 Terabyte (TB) compressed into one image – Big Data in Earth Observation*. URL: https://www.dlr.de/eoc/en/desktopdefault.aspx/tabid-11932/20674_read-48386/ (visited on 05/05/2022).

[3] Abhishek Saxena, Neeta Verma, and Dr K C Tripathi. “A Review Study of Weather Forecasting Using Artificial Neural Network Approach”. In: *International Journal of Engineering Research* 2.11 (2013), pp. 2029–2036.

[4] Qiangqiang Yuan, Huanfeng Shen, Tongwen Li, Zhiwei Li, Shuwen Li, Yun Jiang, Hongzhang Xu, Weiwei Tan, Qianqian Yang, Jiwen Wang, Jianhao Gao, and Liangpei Zhang. “Deep learning in environmental remote sensing: Achievements and challenges”. In: *Remote Sensing of Environment* 241 (May 1, 2020), p. 111716. DOI: 10.1016/j.rse.2020.111716.

[5] Tobias Bolch. “Climate change and glacier retreat in northern Tien Shan (Kazakhstan/Kyrgyzstan) using remote sensing data”. In: *Global and Planetary Change*. Climate Change Impacts on Mountain Glaciers and Permafrost 56.1-2 (Mar. 1, 2007), pp. 1–12. DOI: 10.1016/j.gloplacha.2006.07.009.

[6] Jun Yang, Peng Gong, Rong Fu, Minghua Zhang, Jingming Chen, Shunlin Liang, Bing Xu, Jiancheng Shi, and Robert Dickinson. “The role of satellite remote sensing in climate change studies”. In: *Nature Climate Change* 3.10 (Oct. 2013). Number: 10 Publisher: Nature Publishing Group, pp. 875–883. DOI: 10.1038/nclimate1908.

[7] D. Maktav, F. S. Erbek, and C. Jürgens. “Remote sensing of urban areas”. In: *International Journal of Remote Sensing* 26.4 (Feb. 1, 2005), pp. 655–659. DOI: 10.1080/01431160512331316469.

[8] Thilo Wellmann et al. “Remote sensing in urban planning: Contributions towards ecologically sound policies?” In: *Landscape and Urban Planning* 204 (Dec. 1, 2020), p. 103921. DOI: 10.1016/j.landurbplan.2020.103921.

[9] Dick Kempka and Lisa Lackey. “Using Remote Sensing to Detect and Monitor Land-Cover and Land-Use Change”. In: *Photogrammetric engineering and remote sensing* 60.3 (1994), pp. 331–337.

[10] K.S. Chen, Y.C. Tzeng, C.F. Chen, W.L. Kao, and C.L. Ni. “Classification of multispectral imagery using dynamic learning neural network”. In: *Proceedings of IGARSS’93-IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 1993, 896–898 vol.2. DOI: 10.1109/IGARSS.1993.322194.

[11] Hao Chen, Zipeng Qi, and Zhenwei Shi. “Remote Sensing Image Change Detection With Transformers”. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2021), pp. 1–14. DOI: 10.1109/tgrs.2021.3095166.

[12] John Townshend, Christopher Justice, Wei Li, Charlotte Gurney, and Jim McManus. “Global land cover classification by remote sensing: Present capabilities and future possibilities”. In: *Remote Sensing of Environment* 35.2-3 (Feb. 1, 1991), pp. 243–255. DOI: 10.1016/0034-4257(91)90016-y.

[13] J D White, K C Ryan, C C Key, and S W Running. “Remote Sensing of Forest Fire Severity and Vegetation Recovery”. In: *International Journal of Wildland Fire* 6.3 (1996). Publisher: CSIRO PUBLISHING, p. 125. DOI: 10.1071/wf9960125.

[14] Emilio Chuvieco and Russell G. Congalton. “Application of remote sensing and geographic information systems to forest fire hazard mapping”. In: *Remote Sensing of Environment* 29.2 (Aug. 1, 1989), pp. 147–159. DOI: 10.1016/0034-4257(89)90023-0.

[15] Leila Hashemi-Beni and Asmamaw A. Gebrehiwot. “Flood Extent Mapping: An Integrated Method Using Deep Learning and Region Growing Using UAV Optical Data”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14 (2021), pp. 2127–2135. DOI: 10.1109/jstars.2021.3051873.

[16] Tewodros Tilahun and Wondwosen M. Seyoum. “High-Resolution Mapping of Tile Drainage in Agricultural Fields Using Unmanned Aerial System (UAS)-Based Radiometric Thermal and Optical Sensors”. In: *Hydrology* 8.1 (Dec. 28, 2020), p. 2. DOI: 10.3390/hydrology8010002.

[17] Homin Song, Dong Kook Woo, and Qina Yan. “Detecting subsurface drainage pipes using a fully convolutional network with optical images”. In: *Agricultural Water Management* 249 (Apr. 30, 2021), p. 106791. DOI: 10.1016/j.agwat.2021.106791.

[18] Jan de Leeuw, Anton Vrieling, Apurba Shee, Clement Atzberger, Kiros Hadgu, Chandrashekhar Biradar, Humphrey Keah, and Calum Turvey. “The Potential and Uptake of Remote Sensing in Insurance: A Review”. In: *Remote Sensing* 6.11 (Nov. 2014). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, pp. 10888–10912. DOI: 10.3390/rs61110888.

[19] Maryam Rahnemoonfar, Tashnim Chowdhury, Argho Sarkar, Debvrat Varshney, Masoud Yari, and Robin Roberson Murphy. “FloodNet: A High Resolution Aerial Imagery Dataset for Post Flood Scene Understanding”. In: *IEEE Access* 9 (2021), pp. 89644–89654. DOI: `10.1109/access.2021.3090981`.

[20] Laura Lopez-Fuentes, Joost van de Weijer, Marc Bolanos, and Harald Skinnemoen. “Multi-modal Deep Learning Approach for Flood Detection”. In: *MediaEval* 17 (2017), pp. 13–15.

[21] David So, Quoc Le, and Chen Liang. “The Evolved Transformer”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. PMLR. 2019, pp. 5877–5886.

[22] David Patterson, Joseph Gonzalez, Urs Holzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. “The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink”. In: *Computer* 55.7 (2022), pp. 18–28. DOI: `10.1109/mc.2022.3148714`.

[23] Janis Keuper and Franz-Josef Preundt. “Distributed Training of Deep Neural Networks: Theoretical and Practical Limits of Parallel Scalability”. In: *2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC)*. 2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC). IEEE, Nov. 2016, pp. 19–26. DOI: `10.1109/mlhpc.2016.006`.

[24] Yunyong Ko, Kibong Choi, Jiwon Seo, and Sang-Wook Kim. “An In-Depth Analysis of Distributed Training of Deep Neural Networks”. In: *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS). ISSN: 1530-2075. IEEE, May 2021, pp. 994–1003. DOI: `10.1109/ipdps49936.2021.00108`.

[25] Steven J. Nowlan. “Simplifying Neural Networks by Soft Weight Sharing”. In: *The Mathematics of Generalization*. Num Pages: 22. CRC Press, 1995. Chap. Simplifying Neural Networks by Soft Weight Sharing, pp. 373–394. DOI: `10.1201/9780429492525-13`.

[26] Fatih Köksal, Ethem Alpaydyn, and Günhan Dündar. “Weight Quantization for Multi-layer Perceptrons Using Soft Weight Sharing”. In: *Artificial Neural Networks — ICANN 2001*. Ed. by Georg Dorffner, Horst Bischof, and Kurt Hornik. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2001, pp. 211–216. DOI: `10.1007/3-540-44668-0_30`.

[27] Karen Ullrich, Edward Meeds, and Max Welling. *Soft Weight-Sharing for Neural Network Compression*. May 9, 2017. DOI: `10.48550/arXiv.1702.04008`.

[28] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V. Le. “Can Weight Sharing Outperform Random Architecture Search? An Investigation With TuNAS”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, 2020, pp. 14323–14332. DOI: 10.1109/cvpr42600.2020.01433.

[29] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. “What is the State of Neural Network Pruning?” In: *Proceedings of Machine Learning and Systems* 2 (Mar. 15, 2020), pp. 129–146.

[30] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. *Rethinking the Value of Network Pruning*. Mar. 5, 2019. DOI: 10.48550/arXiv.1810.05270.

[31] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. *Pruning Filters for Efficient ConvNets*. Mar. 10, 2017. DOI: 10.48550/arXiv.1608.08710.

[32] R. Reed. “Pruning algorithms-a survey”. In: *IEEE Transactions on Neural Networks* 4.5 (Sept. 1993). Conference Name: IEEE Transactions on Neural Networks, pp. 740–747. DOI: 10.1109/72.248452.

[33] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. “Fixed Point Quantization of Deep Convolutional Networks”. In: *Proceedings of The 33rd International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228. PMLR, June 11, 2016, pp. 2849–2858.

[34] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. *Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation*. Apr. 20, 2020. DOI: 10.48550/arXiv.2004.09602.

[35] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. “Deep Learning with Low Precision by Half-Wave Gaussian Quantization”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2017, pp. 5918–5926. DOI: 10.1109/cvpr.2017.574.

[36] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. *Mixed Precision Training*. Feb. 15, 2018. DOI: 10.48550/arXiv.1710.03740.

[37] Nhut-Minh Ho and Weng-Fai Wong. “Exploiting half precision arithmetic in Nvidia GPUs”. In: *2017 IEEE High Performance Extreme Computing Conference (HPEC)*. 2017 IEEE High Performance Extreme Computing Conference (HPEC). IEEE, Sept. 2017, pp. 1–7. DOI: 10.1109/hpec.2017.8091072.

[38] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. “Highly Scalable Deep Learning Training System with Mixed-Precision: Training ImageNet in Four Minutes”. In: *International Conference on Machine Learning*. International Conference on Machine Learning. PMLR. 2018, pp. 4904–4916. DOI: 10.48550/arXiv.1807.11205.

[39] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. Mar. 9, 2015. DOI: 10.48550/arXiv.1503.02531.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. “Attention Is All You Need”. In: *Advances in neural information processing systems* 30 (Dec. 5, 2017). DOI: 10.48550/arXiv.1706.03762.

[41] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1997). Conference Name: Neural Computation, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[42] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. Dec. 11, 2014. DOI: 10.48550/arXiv.1412.3555.

[43] Francois Spoto, Omar Sy, Paolo Laberinti, Philippe Martimort, Valerie Fernandez, Olivier Colin, Bianca Hoersch, and Aime Meygret. “Overview Of Sentinel-2”. In: *2012 IEEE International Geoscience and Remote Sensing Symposium*. IGARSS 2012 - 2012 IEEE International Geoscience and Remote Sensing Symposium. Munich, Germany: IEEE, July 2012, pp. 1707–1710. DOI: 10.1109/igarss.2012.6351195.

[44] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. *A Structured Self-attentive Sentence Embedding*. Mar. 8, 2017. DOI: 10.48550/arXiv.1703.03130.

[45] David E. Goldberg and Kalyanmoy Deb. “A Comparative Analysis of Selection Schemes Used in Genetic Algorithms”. In: *Foundations of Genetic Algorithms*. Ed. by GREGORY J. E. Rawlins. Vol. 1. Elsevier, Jan. 1, 1991, pp. 69–93. DOI: 10.1016/b978-0-08-050684-5.50008-2.

[46] Ioannis Papoutsis, Nikolaos-Ioannis Bountos, Angelos Zavras, Dimitrios Michail, and Christos Tryfonopoulos. “Efficient deep learning models for land cover image classification”. In: *Computing Research Repository (CoRR)* (Nov. 17, 2021). DOI: 10.48550/arXiv.2111.09451.

[47] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Computing Research Repository (CoRR)* (May 24, 2019). DOI: 10.48550/arXiv.1810.04805.

[48] Wilson L. Taylor. ""Cloze Procedure": A New Tool for Measuring Readability". In: *Journalism Quarterly* 30.4 (Sept. 1, 1953). Publisher: SAGE Publications, pp. 415–433. DOI: 10.1177/107769905303000401.

[49] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. "Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Proceedings of the IEEE International Conference on Computer Vision. IEEE, 2015, pp. 19–27. DOI: 10.1109/iccv.2015.11.

[50] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter". In: *Computing Research Repository (CoRR)* (Feb. 29, 2020). DOI: 10.48550/arXiv.1910.01108.

[51] M.W. Newhouse and R.T. Hanson. *Three-dimensional measurements of flow in uncased wells completed in basalt, Mountain Home Air Force Base, Idaho, March 2000*. Tech. rep. US Geological Survey, 2002. DOI: 10.3133/wri014259.

[52] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. "TinyBERT: Distilling BERT for Natural Language Understanding". In: *Computing Research Repository (CoRR)* (Oct. 15, 2020). DOI: 10.48550/arXiv.1909.10351.

[53] huggingface.co. *huawei-noah/TinyBERT_General_4L_312D · Hugging Face*. URL: https://huggingface.co/huawei-noah/TinyBERT_General_4L_312D (visited on 06/27/2022).

[54] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. "What Does BERT Look at? An Analysis of BERT's Attention". In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, June 10, 2019. DOI: 10.18653/v1/w19-4828.

[55] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Well-Read Students Learn Better: On the Importance of Pre-training Compact Models". In: *Computing Research Repository (CoRR)* (Sept. 25, 2019). DOI: 10.48550/arXiv.1908.08962.

[56] huggingface.co. *prajjwal1/bert-tiny · Hugging Face*. URL: https://huggingface.co/prajjwal1/bert-tiny (visited on 06/27/2022).

[57] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. "MobileBERT: A Compact Task-Agnostic BERT for Resource-Limited Devices". In: *Computing Research Repository (CoRR)* (Apr. 14, 2020). DOI: 10.48550/arXiv.2004.02984.

[58] huggingface.co. *google/mobilebert-uncased · Hugging Face*. URL: https://huggingface.co/google/mobilebert-uncased (visited on 06/27/2022).

[59] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations”. In: *Computing Research Repository (CoRR)* (Feb. 8, 2020). DOI: 10.48550/arXiv.1909.11942.

[60] huggingface.co. *albert-base-v2 · Hugging Face*. URL: https://huggingface.co/albert-base-v2 (visited on 06/27/2022).

[61] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer. “Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources”. In: *IEEE Geoscience and Remote Sensing Magazine* 5.4 (Dec. 2017), pp. 8–36. DOI: 10.1109/mgrs.2017.2762307.

[62] Lei Ma, Yu Liu, Xueliang Zhang, Yuanxin Ye, Gaofei Yin, and Brian Alan Johnson. “Deep learning in remote sensing applications: A meta-analysis and review”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 152 (June 2019), pp. 166–177. DOI: 10.1016/j.isprsjprs.2019.04.015.

[63] Xuelei Chen. “Generative Adversarial Networks for Content-based Retrieval of Multispectral Images”. PhD thesis. Berlin: Technische Universität Berlin, Sept. 2019. 87 pp.

[64] Ahmed Elshamli, Graham W. Taylor, and Shawki Areibi. “Multisource Domain Adaptation for Remote Sensing Using Deep Neural Networks”. In: *IEEE Transactions on Geoscience and Remote Sensing* 58.5 (May 2020), pp. 3328–3340. DOI: 10.1109/tgrs.2019.2953328.

[65] Devis Tuia, Claudio Persello, and Lorenzo Bruzzone. “Domain Adaptation for the Classification of Remote Sensing Data: An Overview of Recent Advances”. In: *IEEE Geoscience and Remote Sensing Magazine* 4.2 (June 2016), pp. 41–57. DOI: 10.1109/mgrs.2016.2548504.

[66] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Apr. 8, 2009.

[67] Mia Xu Chen et al. “The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. International Conference on Machine Learning. ISSN: 2640-3498. Association for Computational Linguistics, July 3, 2018, pp. 4055–4064. DOI: 10.18653/v1/p18-1008.

[68] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. *Scaling Autoregressive Video Models*. Feb. 10, 2020. DOI: 10.48550/arXiv.1906.02634.

[69] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. *Axial Attention in Multidimensional Transformers*. Dec. 20, 2019. DOI: 10.48550/arXiv.1912.12180.

[70] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. “Non-local Neural Networks”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2018, pp. 7794–7803. DOI: `10.1109/cvpr.2018.00813`.

[71] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. *Generating Long Sequences with Sparse Transformers*. Apr. 23, 2019. DOI: `10.48550/arXiv.1904.10509`.

[72] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. *On the Relationship between Self-Attention and Convolutional Layers*. Jan. 10, 2020. DOI: `10.48550/arXiv.1911.03584`.

[73] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. “The Open Images Dataset V4. Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale”. In: *Int. J. Comput. Vision* 128.7 (June 3, 2021), pp. 1956–1981. DOI: `10.1007/s11263-020-01316-z`.

[74] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009 IEEE Conference on Computer Vision and Pattern Recognition. ISSN: 1063-6919. IEEE, June 2009, pp. 248–255. DOI: `10.1109/cvpr.2009.5206848`.

[75] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Proceedings of the IEEE International Conference on Computer Vision. IEEE, 2017, pp. 843–852. DOI: `10.1109/iccv.2017.97`.

[76] Sachin Mehta and Mohammad Rastegari. “MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer”. In: *Computing Research Repository (CoRR)* (Oct. 5, 2021). DOI: `10.48550/arXiv.2110.02178`.

[77] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. “MobileNetV2: Inverted Residuals and Linear Bottlenecks”. In: *Computing Research Repository (CoRR)* (Mar. 21, 2019). DOI: `10.48550/arXiv.1801.04381`.

[78] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. “Mobile-Former: Bridging MobileNet and Transformer”. In: *Computing Research Repository (CoRR)* (Aug. 12, 2021). DOI: `10.48550/arXiv.2108.05895`.

[79] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. “CvT: Introducing Convolutions to Vision Transformers”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Mar. 29, 2021. DOI: `10.1109/iccv48922.2021.00009`.

[80] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. *Dynamic ReLU*. Aug. 5, 2020. DOI: `10.48550/arXiv.2003.10027`.

[81] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. “CSWin Transformer: A General Vision Transformer Backbone with Cross-Shaped Windows”. In: *Computing Research Repository (CoRR)* (Jan. 9, 2022). DOI: `10.48550/arXiv.2107.00652`.

[82] Asher Trockman and J. Zico Kolter. “Patches Are All You Need?” In: *Computing Research Repository (CoRR)* (Jan. 24, 2022). DOI: `10.48550/arXiv.2201.09792`.

[83] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. *MLP-Mixer: An all-MLP Architecture for Vision*. June 11, 2021. DOI: `10.48550/arXiv.2105.01601`.

[84] Ross Wightman. *PyTorch Image Models*. `https://github.com/rwightman/pytorch-image-models`. May 24, 2022.

[85] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. “VQA: Visual Question Answering”. In: *Int. J. Comput. Vision* 123.1 (Oct. 26, 2016), pp. 4–31. DOI: `10.1007/s11263-016-0966-6`.

[86] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. *Zero-Shot Text-to-Image Generation*. Feb. 26, 2021. DOI: `10.48550/arXiv.2102.12092`.

[87] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. Apr. 12, 2022. DOI: `10.48550/arXiv.2204.06125`.

[88] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. *Learning Transferable Visual Models From Natural Language Supervision*. Feb. 26, 2021. DOI: `10.48550/arXiv.2103.00020`.

[89] Chitwan Saharia et al. *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding*. May 23, 2022. DOI: `10.48550/arXiv.2205.11487`.

[90] Liunian Harold Li, Patrick H. Chen, Cho-Jui Hsieh, and Kai-Wei Chang. "Efficient Contextual Representation Learning With Continuous Outputs". In: *Transactions of the Association for Computational Linguistics* 7 (Aug. 9, 2019), pp. 611–624. DOI: 10.1162/tacl_a_00289.

[91] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

[92] Jean-Baptiste Alayrac et al. *Flamingo: A Visual Language Model for Few-Shot Learning*. Apr. 29, 2022. DOI: 10.48550/arXiv.2204.14198.

[93] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. "Perceiver: General Perception with Iterative Attention". In: *Proceedings of the 38th International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 2640-3498. PMLR, July 1, 2021, pp. 4651–4664.

[94] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. *CoCa: Contrastive Captioners are Image-Text Foundation Models*. June 13, 2022. DOI: 10.48550/arXiv.2205.01917.

[95] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. "LiT: Zero-shot Transfer with Locked-image text Tuning". In: *Computing Research Repository (CoRR)* (Mar. 25, 2022). DOI: 10.48550/arXiv.2111.07991.

[96] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. "Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision". In: *International Conference on Machine Learning*. International Conference on Machine Learning. PMLR. 2021, pp. 4904–4916.

[97] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. "ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

[98] Sylvain Lobry, Diego Marcos, Jesse Murray, and Devis Tuia. "RSVQA: Visual Question Answering for Remote Sensing Data". In: *IEEE Transactions on Geoscience and Remote Sensing* 58.12 (Dec. 2020), pp. 8555–8566. DOI: 10.1109/tgrs.2020.2988782.

[99] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2016, pp. 770–778. DOI: 10.1109/cvpr.2016.90.

[100] Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. “Skip-Thought Vectors”. In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015.

[101] Sylvain Lobry, Begum Demir, and Devis Tuia. “RSVQA Meets Bigearthnet: A New, Large-Scale, Visual Question Answering Dataset for Remote Sensing”. In: *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*. IGARSS 2021 - 2021 IEEE International Geoscience and Remote Sensing Symposium. Brussels, Belgium: IEEE, July 11, 2021, pp. 1218–1221. DOI: 10.1109/igarss47720.2021.9553307.

[102] Gencer Sumbul, Arne de Wall, Tristan Kreuziger, Filipe Marcelino, Hugo Costa, Pedro Benevides, Mario Caetano, Begum Demir, and Volker Markl. “BigEarthNet-MM: A Large-Scale, Multimodal, Multilabel Benchmark Archive for Remote Sensing Image Classification and Retrieval [Software and Data Sets]”. In: *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium* 9.3 (July 2019), pp. 174–180. DOI: 10.1109/mgrs.2021.3089174.

[103] Christel Chappuis, Sylvain Lobry, Benjamin Kellenberger, Bertrand Le Saux, and Devis Tuia. “How to find a good image-text embedding for remote sensing visual question answering?” In: *Computing Research Repository (CoRR)* (Sept. 24, 2021). DOI: 10.48550/arXiv.2109.11848.

[104] Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. *Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding*. Sept. 23, 2016. DOI: 10.48550/arXiv.1606.01847.

[105] Hedi Ben-younes, Remi Cadene, Matthieu Cord, and Nicolas Thome. “MUTAN: Multimodal Tucker Fusion for Visual Question Answering”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Proceedings of the IEEE International Conference on Computer Vision. IEEE, 2017, pp. 2612–2620. DOI: 10.1109/iccv.2017.285.

[106] Christel Chappuis, Valérie Zermatten, Sylvain Lobry, Bertrand Le Saux, and Devis Tuia. “Prompt-RSVQA: Prompting Visual Context to a Language Model for Remote Sensing Visual Question Answering”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022, pp. 1372–1381.

[107] Xiangtao Zheng, Binqiang Wang, Xingqian Du, and Xiaoqiang Lu. “Mutual Attention Inception Network for Remote Sensing Visual Question Answering”. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–14. DOI: 10.1109/tgrs.2021.3079918.

[108] Yi Yang and Shawn Newsam. "Bag-of-visual-words and spatial extensions for land-use classification". In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '10*. GIS '10. New York, NY, USA: ACM Press, Nov. 2, 2010, pp. 270–279. DOI: 10.1145/1869790.1869829.

[109] Fan Zhang, Bo Du, and Liangpei Zhang. "Saliency-Guided Unsupervised Feature Learning for Scene Classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 53.4 (Apr. 2015). Conference Name: IEEE Transactions on Geoscience and Remote Sensing, pp. 2175–2184. DOI: 10.1109/tgrs.2014.2357078.

[110] Yuanlin Zhang, Yuan Yuan, Yachuang Feng, and Xiaoqiang Lu. "Hierarchical and Robust Convolutional Neural Network for Very High-Resolution Remote Sensing Object Detection". In: *IEEE Transactions on Geoscience and Remote Sensing* 57.8 (Aug. 2019). Conference Name: IEEE Transactions on Geoscience and Remote Sensing, pp. 5535–5548. DOI: 10.1109/tgrs.2019.2900302.

[111] Gui-Song Xia, Jingwen Hu, Fan Hu, Baoguang Shi, Xiang Bai, Yanfei Zhong, Liangpei Zhang, and Xiaoqiang Lu. "AID: A Benchmark Data Set for Performance Evaluation of Aerial Scene Classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 55.7 (July 2017). Conference Name: IEEE Transactions on Geoscience and Remote Sensing, pp. 3965–3981. DOI: 10.1109/tgrs.2017.2685945.

[112] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. "DOTA: A Large-Scale Dataset for Object Detection in Aerial Images". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2018, pp. 3974–3983. DOI: 10.1109/cvpr.2018.00418.

[113] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. Apr. 10, 2015. DOI: 10.48550/arXiv.1409.1556.

[114] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. Sept. 6, 2013. DOI: 10.48550/arXiv.1301.3781.

[115] Zhenghang Yuan, Lichao Mou, Qi Wang, and Xiao Xiang Zhu. "From Easy to Hard: Learning Language-Guided Curriculum for Visual Question Answering on Remote Sensing Data". In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022). Conference Name: IEEE Transactions on Geoscience and Remote Sensing, pp. 1–11. DOI: 10.1109/tgrs.2022.3173811.

[116] Matthieu Molinier, Jukka Miettinen, Dino Ienco, Shi Qiu, and Zhe zhu. *Optical Satellite Image Time Series Analysis for Environment Applications: From Classical Methods to Deep Learning and Beyond*. Dec. 12, 2021. doi: 10.1002/9781119882299.ch4.

[117] Maxim Neumann, Andre Susano Pinto, Xiaohua Zhai, and Neil Houlsby. "Training General Representations for Remote Sensing Using in-Domain Knowledge". In: *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 2020, pp. 6730–6733. doi: 10.1109/igarss39084.2020.9324501.

[118] Vladimir Risojević and Vladan Stojnić. "The Role of Pre-Training in High-Resolution Remote Sensing Scene Classification". In: *Computing Research Repository (CoRR)* (Dec. 17, 2021). doi: 10.48550/arXiv.2111.03690.

[119] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[120] André Mora, Tiago Santos, Szymon Łukasik, João Silva, António Falcão, José Fonseca, and Rita Ribeiro. "Land Cover Classification from Multispectral Data Using Computational Intelligence Tools: A Comparative Study". In: *Information* 8.4 (Nov. 15, 2017), p. 147. doi: 10.3390/info8040147.

[121] Daniela I. Moody, Steven P. Brumby, Joel C. Rowland, and Garrett L. Altmann. "Land cover classification in multispectral imagery using clustering of sparse approximations over learned feature dictionaries". In: *Journal of Applied Remote Sensing* 8.1 (Dec. 2014). Publisher: SPIE, p. 084793. doi: 10.1117/1.jrs.8.084793.

[122] Gencer Sumbul, Arne de Wall, Tristan Kreuziger, Filipe Marcelino, Hugo Costa, Pedro Benevides, Mario Caetano, Begum Demir, and Volker Markl. "BigEarthNet-MM: A Large-Scale, Multimodal, Multilabel Benchmark Archive for Remote Sensing Image Classification and Retrieval [Software and Data Sets]". In: *IEEE Geoscience and Remote Sensing Magazine* 9.3 (Sept. 2021), pp. 174–180. doi: 10.1109/mgrs.2021.3089174.

[123] Dirk Geudtner, Ramon Torres, Paul Snoeij, Malcolm Davidson, and Bjorn Rommen. "Sentinel-1 System capabilities and applications". In: *2014 IEEE Geoscience and Remote Sensing Symposium*. IGARSS 2014 - 2014 IEEE International Geoscience and Remote Sensing Symposium. Quebec City, QC: IEEE, July 2014, pp. 1457–1460. doi: 10.1109/igarss.2014.6946711.

[124] Peg Shippert. "Why Use Hyperspectral Imagery?" In: *Photogrammetric engineering and remote sensing* 70.4 (2004), pp. 377–396.

[125] Ayan Chakrabarti and Todd Zickler. “Statistics of real-world hyperspectral images”. In: *CVPR 2011*. CVPR 2011. ISSN: 1063-6919. IEEE, June 2011, pp. 193–200. DOI: 10.1109/cvpr.2011.5995660.

[126] Li Guo, Nesrine Chehata, Clément Mallet, and Samia Boukir. “Relevance of airborne lidar and multispectral image data for urban scene classification using Random Forests”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 66.1 (Jan. 1, 2011), pp. 56–66. DOI: 10.1016/j.isprsjprs.2010.08.007.

[127] Samuli Junttila, Roope Näsi, Niko Koivumäki, Mohammad Imangholiloo, Ninni Saarinen, Juha Raisio, Markus Holopainen, Hannu Hyyppä, Juha Hyyppä, Päivi Lyytikäinen-Saarenmaa, Mikko Vastaranta, and Eija Honkavaara. “Multispectral Imagery Provides Benefits for Mapping Spruce Tree Decline Due to Bark Beetle Infestation When Acquired Late in the Season”. In: *Remote Sensing* 14.4 (Jan. 2022). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 909. DOI: 10.3390/rs14040909.

[128] Stefan Thomas, Matheus Thomas Kuska, David Bohnenkamp, Anna Brugger, Elias Alisaac, Mirwaes Wahabzada, Jan Behmann, and Anne-Katrin Mahlein. “Benefits of hyperspectral imaging for plant disease detection and plant protection: A technical perspective”. In: *Journal of Plant Diseases and Protection* 125.1 (Feb. 1, 2018), pp. 5–20. DOI: 10.1007/s41348-017-0124-6.

[129] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. Jan. 4, 2019. DOI: 10.48550/arXiv.1711.05101.

[130] William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Version 1.10. https://www.pytorchlightning.ai. Mar. 30, 2019.

[131] Marina Sokolova and Guy Lapalme. “A systematic analysis of performance measures for classification tasks”. In: *Information Processing & Management* 45.4 (July 1, 2009), pp. 427–437. DOI: 10.1016/j.ipm.2009.03.002.

[132] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. *LocalViT: Bringing Locality to Vision Transformers*. Apr. 12, 2021. DOI: 10.48550/arXiv.2104.05707.

[133] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. “Sigmoid-weighted linear units for neural network function approximation in reinforcement learning”. In: *Neural Networks*. Special issue on deep reinforcement learning 107 (Nov. 1, 2018), pp. 3–11. DOI: 10.1016/j.neunet.2017.12.012.

[134] Michael Titterington. *Neural Networks*. Aug. 9, 2020. DOI: 10.1002/9781118445112.stat00448.

[135] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feed-forward networks are universal approximators". In: *Neural Networks* 2.5 (Jan. 1989), pp. 359–366. DOI: 10.1016/0893-6080(89)90020-8.

[136] sentinels.copernicus.eu. *Spatial - Resolutions - Sentinel-2 MSI - User Guides - Sentinel Online - Sentinel Online*. URL: https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-2-msi/resolutions/spatial (visited on 04/28/2022).

[137] sentinels.copernicus.eu. *Radiometric - Resolutions - Sentinel-2 MSI - User Guides - Sentinel Online - Sentinel Online*. URL: https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-2-msi/resolutions/radiometric (visited on 04/28/2022).

7 Appendix

Table 8: Detailed model dimensions comparison between BERT [47] and teacher and student of MobileBERT [57, tab. 1].

			BERT		MobileBERT	
			BERT$_{LARGE}$	BERT$_{BASE}$	IB-BERT$_{LARGE}$ Teacher	MobileBERT$_{Tiny}$ Student
embedding		$h_{embedding}$	1024	768	128	
			-	-	3-convolution	
		h_{inter}	1024	768	512	
body	Linear	h_{input}			512	512
		h_{output}			1024	128
	MHA	h_{input}	1024	768	512	128
		#Head	16	12	4	4
		h_{output}	1024	768	1024	128
	FFN	h_{input}	1024	768	1024	128
		h_{FFN}	4096	3072	4096	512
		h_{output}	1024	768	1024	128 ×2
	Linear	h_{input}			1024	128
		h_{output}			512	512
			× 24	× 12	× 24	× 24

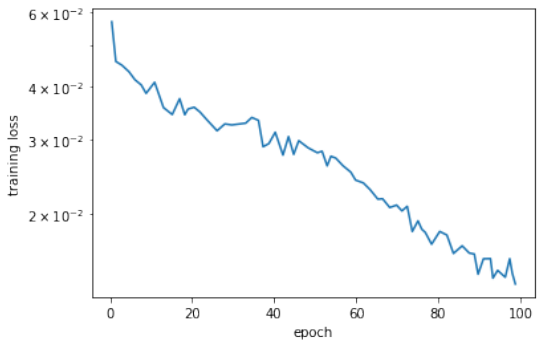


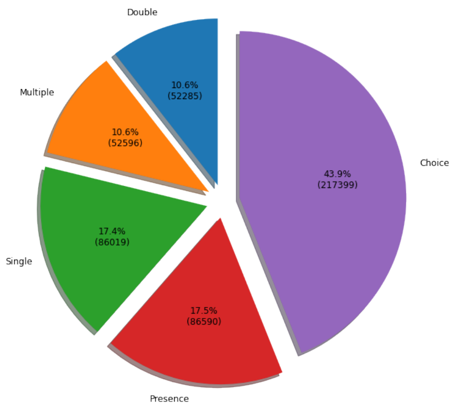
Figure 33: Training loss of the ConvMixer Network. Displayed without validation loss for scaling.

Table 9: Wavelength and Bandwidth of the 13 Sentinel-2 Bands. Sentinel-2 satellites have two slightly different configurations (S2A and S2B) which have minimal different specifications. All bands have a radiometric resolution of 12 bit. [136]

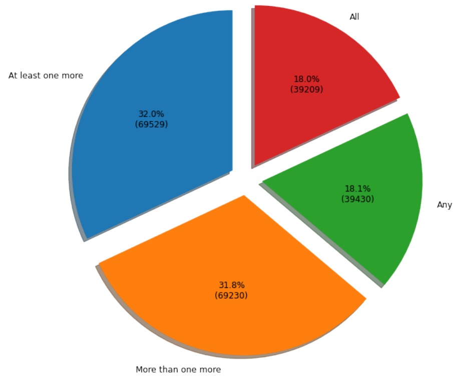
Spacial Resolution	Band Name	Central wavelength (nm)		Bandwidth (nm)	
		S2A	S2B	S2A	S2B
10m	B02	492	492	66	66
	B03	560	559	36	36
	B04	665	665	31	31
	B08	833	833	106	106
20m	B05	704	704	15	16
	B06	742	739	15	15
	B07	783	780	20	20
	B08a	865	864	21	22
	B11	1614	1610	91	94
	B12	2202	2186	175	185
60m	B01	443	442	21	21
	B09	945	943	20	21
	B10	1374	1377	31	30

Table 10: Use cases of the 13 Sentinel-2 Bands. Different spacial resolutions are assigned to different use cases like Cloud Discriminations (CDs) or Cirrus cloud detection. [137]

Spacial Resolution	Band Name	Wavelength Range	Use case
10m	B02	VIS	Blue light
	B03	VIS	Green light
	B04	VIS	Red light
	B08	NIR	HR-NIR for LCC
20m	B05	NIR	Vegetation Red-edge
	B06	NIR	Vegetation Red-edge
	B07	NIR	Vegetation Red-edge
	B08a	NIR	Vegetation Red-edge
	B11	SWIR	Snow& Ice Detection/CD
	B12	SWIR	Snow& Ice Detection/CD
60m	B01	VIS	Aerosols
	B09	NIR	Water-vapor
	B10	SWIR	Cirrus clouds

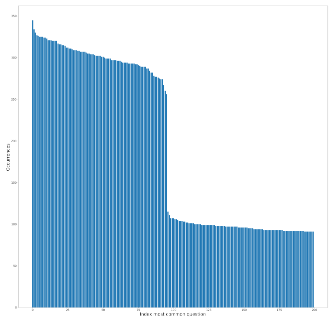


(a) Question type distribution for the main types *Choice*, *Presence*, *Singe*, *Double* and *Multiple* for the validation set.

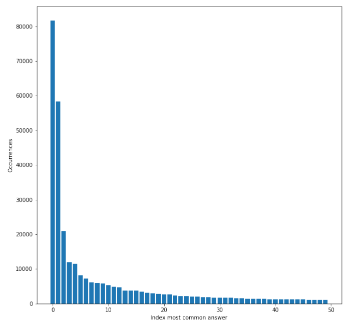


(b) Question sub-type distribution for the main type *Choice* for the validation set.

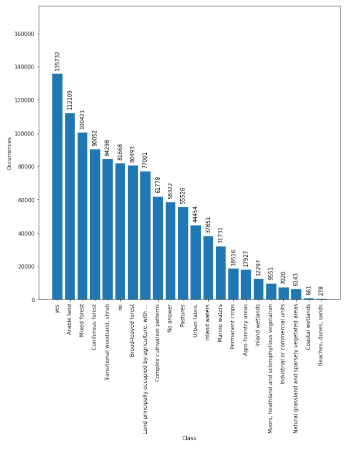
Figure 34: Question type distributions for the validation set.



(a) Absolute frequencies of most common questions for the validation set. Questions to the left of the drop around index 95 are about *Presence*. Questions to the right of the drop are about other types.

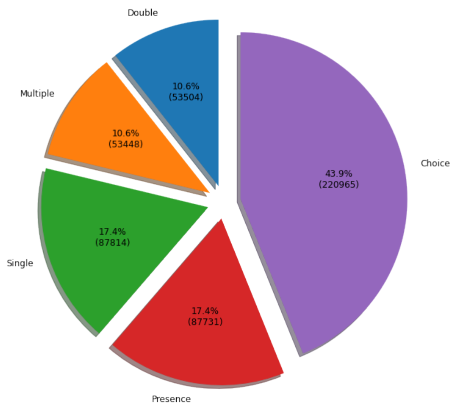


(b) Absolute frequencies of most common answer combinations for the validation set. Most common combinations are “No”, “No Answer”, and “Marine waters” in all sets.

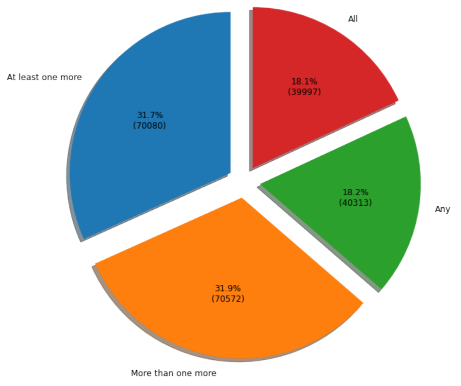


(c) Absolute frequencies of classes in the validation set. Class name shorted like in the training set.

Figure 35: Absolute frequencies of *n* most common Questions and Answers in the validation set.

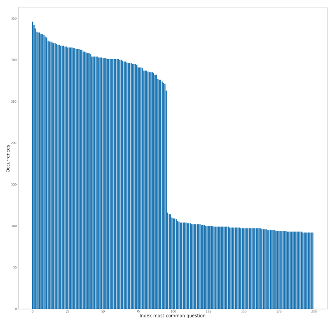


(a) Question type distribution for the main types *Choice*, *Presence*, *Singe*, *Double* and *Multiple* for the test set.

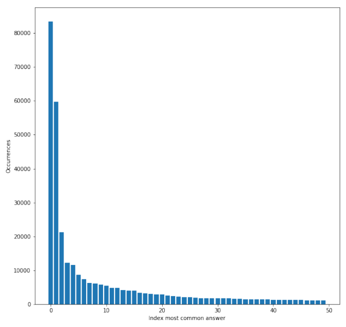


(b) Question sub-type distribution for the main type *Choice* for the test set.

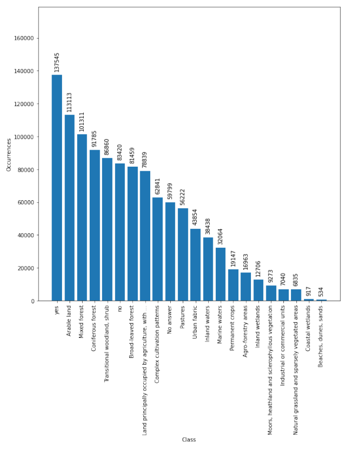
Figure 36: Question type distributions for the test set.



(a) Absolute frequencies of most common questions for the test set. Questions to the left of the drop around index 95 are about *Presence*. Questions to the right of the drop are about other types.



(b) Absolute frequencies of most common answer combinations for the test set. Most common combinations are “No”, “No Answer”, and “Marine waters” in all sets.



(c) Absolute frequencies of classes in the test set. Class name shorted like in the training set.

Figure 37: Absolute frequencies of *n* most common Questions and Answers in the test set.