VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTATIONAL AND DATA MODELING

Master's final thesis

Vision large language models for satellite imagery captioning: a Geolinguistics approach

Done by:
Linas Ramanauskas

Supervisor:
Valentas Gružauskas

Vilnius
2024

Contents

Abstract

This research endeavors to investigate, document, and implement a system aimed at the recognition of geographical areas using satellite imagery. The primary objective is to explore the potential of artificial intelligence (AI) image recognition techniques in describing and identifying areas depicted in satellite images. By leveraging AI technologies, particularly large-scale language models and computer vision algorithms, the research aims to develop a search/recommendation system capable of matching the user-given prompt with corresponding satellite images in the dataset. The result from the experiments of pretrained data reveals that utilizing a common open-source dataset and a pre-trained CLIP model can match the captioning with recommended images fairly well. To adopt a longer, descriptive captioning (e.g. from Wikipedia) for photos, we will be required to fine-tune a model with a custom dataset, in our case it was decided to gather descriptions about Lithuanian churches. The Dataset had to be constructed manually, since there was no way to automate and extract data specifically about the visuals of the church and its surroundings based on Wikipedia descriptions. The images were extracted from google maps API, thanks to "Genčių genealogija" it was a possibility to get geo points of the churches and extract exact images based on those points. The document also describes the different methods of approaching such limitations working with long descriptive data and providing two solutions to overcome long descriptions, such as token pooling and extended context lengths. The experiments demonstrated that fine-tuning CLIP with an increased context length outperformed token pooling, achieving higher cosine similarity scores (80% on the custom dataset and 86% on RSICD) and improved attention precision for complex queries. Token pooling, while computationally efficient, resulted in dispersed attention and struggled with nuanced architectural descriptions, achieving only 61% cosine similarity. Dataset augmentation further enhanced generalization, with a smaller batch size of 32 yielding the best results (83% cosine similarity). Overall, the increased context length approach provided better alignment and understanding, highlighting the importance of longer textual inputs and dataset quality, as well as the importance of a larger training data for fine-tuning.

Keywords: Search engine, Recommendation sytstem, LLM, Image-text, Vector databse, Semantic search, similarity, CLIP

Anotacija

Šis tyrimas siekia ištirti, dokumentuoti ir įgyvendinti sistemą, skirtą geografinių vietovių atpažinimui naudojant palydovines nuotraukas. Pagrindinis tikslas – ištirti dirbtinio intelekto (DI) vaizdų atpažinimo technologijų potencialą aprašant ir identifikuojant palydovinėse nuotraukose vaizduojamas vietoves. Pasitelkus DI technologijas, ypač didelio masto kalbos modelius ir kompiuterinės regos algoritmus, siekiama sukurti paieškos ir rekomendacijų sistemą, galinčią suderinti vartotojo pateiktą užklausą su atitinkamomis palydovinėmis nuotraukomis duomenų rinkinyje. Eksperimentų su iš anksto apmokytais duomenimis rezultatai atskleidė, kad naudojant bendrą atvirojo kodo duomenų rinkinį ir iš anksto apmokytą CLIP modelį, galima gana tiksliai suderinti antraštes su rekomenduojamomis nuotraukomis. Norint pritaikyti ilgesnes, išsamias antraštes (pvz., iš Vikipedijos“) nuotraukoms, reikėtų tikslinti modelį naudojant pasirinktą duomenų rinkinį. Mūsų atveju buvo nuspręsta rinkti aprašymus apie Lietuvos bažnyčias. Duomenų rinkinį reikėjo sukurti rankiniu būdu, nes nebuvo galimybės automatizuoti ir išgauti duomenų, susijusių tik su bažnyčių vizualiais aspektais ir jų aplinka, remiantis Vikipedijos“ aprašymais. Nuotraukos buvo surinktos naudojant Google Maps API“, o dėl Genčių genealogijos“ buvo įmanoma gauti bažnyčių geografinius taškus ir išgauti tikslias nuotraukas pagal tuos taškus. Dokumente taip pat aprašomi skirtingi metodai, kaip įveikti apribojimus, susijusius su ilgais aprašomaisiais duomenimis, pateikiant du sprendimus: token pooling (žymeklių sujungimas) ir konteksto ilgio padidinimas. Eksperimentai parodė, kad CLIP modelio tikslinimas su padidintu konteksto ilgiu pranoko žymeklių sujungimą, pasiekdamas aukštesnius kosinuso panašumo rodiklius (80 % su pasirinktiniu duomenų rinkiniu ir 86 % su RSICD) bei tikslesnį dėmesio paskirstymą sudėtingoms užklausoms. Nors žymeklių sujungimas buvo skaičiavimo požiūriu efektyvus, jis nulėmė išsklaidytą dėmesį ir sunkumus apibūdinant niuansuotas architektūrines detales, pasiekdamas tik 61 % kosinį panašumą. Duomenų rinkinio praplėtimas dar labiau pagerino generalizaciją, o mažesnis particijos (ang. batch) dydis (32) davė geriausius rezultatus (83 % kosinuso panašumas). Apskritai padidinto konteksto ilgio požiūris užtikrino geresnį suderinamumą ir supratimą, pabrėžiant ilgesnių tekstinių duomenų rinkinio kokybės ir didesnio mokymosi duomenų rinkinio svarbą modelio tikslinimui.

Raktažodžiai: Paieškos sistema, rekomendavimo sistema, LLM, vaizdas-tekstas, vektorinė duomenų bazė, semantinė paieška, panašumas, CLIP

Introduction

In recent years, the convergence of computer vision and natural language processing has led to remarkable advancements in various fields, particularly in the domain of image understanding and captioning. Vision Large Language Models (LLMs) represent a significant milestone in this convergence, offering a powerful framework for generating textual descriptions of visual content. While initially developed to process and generate text-based data, the application of LLMs to visual inputs, particularly satellite imagery, has gained substantial interest due to its potential to revolutionize geospatial analysis, disaster response, urban planning, environmental monitoring, and many other fields. The land use/land cover studies contribute to urban region and infrastructure planning, optimizations of agriculture, environmentally-friendly management of the resources on earth, detection of climate changes, biodiversity protection, management of disasters, and land degradation monitoring [1].

The motivation behind employing Vision LLMs for satellite imagery captioning stems from the inherent complexity of understanding and interpreting vast amounts of visual data captured by satellites. The effectiveness of using large language models via prompting on vision-language tasks, emphasizing the importance of verbalizing visual content through image captioning [2]. Traditional computer vision approaches often struggle with the nuances and contextual information present in satellite images, limiting their effectiveness in contectualizing meaningful descriptions. Furthermore, manual annotation of satellite imagery is labor-intensive, time-consuming, and often impractical due to the sheer volume of available data. Vision LLMs offer a promising solution to these challenges by leveraging their ability to learn complex patterns and relationships from large-scale datasets, thereby automating the process of image understanding and searching.

In terms of geolinguistics we might be working with a vast amount of data presented in an image, and to comply with very human like verbosity it is necessary to train a model with such a dataset. Not only that, the models need to somehow accept the length of the textual description. Luckily, there are a couple of methods to overcome long descriptions for LLM's like CLIP. Two of the mentioned being the token pooling strategy, and the other, increasing the context length of the model. The token pooling method, although not as effective in generalizing and taking a tole on the accuracy, it is a lot more computationally efficient, and the increased context length approach will result in a slower computational time but most likely better at preserving data on long text inputs. Comparing these methods will allow us to create an experiment to test how viable these approaches are.

The objective of this thesis is to explore, document, and attempt an implementation of a system that will help recognize an area based on satellite imagery. The idea is to utilize artificial intelligence (specifically CLIP) image recognition to associate an area or a geographical object and match the closest match from the dataset. This entails developing a search/recommendation system, a form of an engine that will be able to retrieve the closest match in the dataset by the given prompt about geolocational elements that are described. By harnessing the capabilities of Vision LLMs (specifically CLIP), we aim to address the difference of traditional methods and see if it is even possible to construct a very niche case for a CLIP model and test out it's capabilities. To achieve this goal a custom dataset was constructed manually and a series of experiments are performed, utilizing a selected data set and algorithms, transfer learning and fine-tuning. The experiments reveal the performance difference of applying different strategies (increased context length and token pooling) of adapting long descriptive text for specific satellite images, the difference between custom created dataset and a publicly available dataset and how our finetuned CLIP

models compare with similar publicly available CLIP model. Inspecting the training process, parameter optimizations, various criteria, and combinations can determine the best results (based on Loss value, average co similarity, attention focus and top 100 testing). This, combined with findings and the results of several experiments, can test satelite image retrieval on long geolinguistical descriptions possible.

The tasks of this final master thesis project are:

- To analyze literature, including related work, projects, semantic search and traditional search system comparison relevant to the subject of semantic search, and satellite imagery captioning.
- To create methodology for semantic search system, which includes, constructing a custom dataset with long descriptions, selecting a publicly available dataset for comparison, algorithms, hyper parameter fine-tuning.
- To perform experiments - utilizing and replicating a publiclally known CLIP fine-tuned model with it's available data, training CLIP models for various scenarios and testing different parameters and datasets, displaying cosimilarity heat maps, manual testing with top 100 images and attention heatmap testing.

The result of the master thesis is a CLIP model for semantic search, fine-tuned for the retrieval task of Lithuanian churches and trained with Wikipedia descriptions of the churches.

Applicability of the Study

Broader Applications: This research has implications for geospatial analysis, urban planning and environmental monitoring, where satellite imagery plays a crucial role. Searching for specific geolocational areas based on human like verbosity.

Fine-Tuning for Niche Domains: The methodology demonstrates the viability of fine-tuning models for specific use cases with small, custom datasets, such as manually annotated satellite images of Lithuanian churches.

This thesis is an extension of this paper author's research study of the same title. Main parts of the literature review and methodology where utilized, figures, tables, references.

1 Literature Review

1.1 Geolinguistics

Geolinguistics is an interdisciplinary field that explores the relationship between language and geographical spaces, considering cultural, historical, and sociopolitical contexts. At its core, geolinguistics investigates how language evolves, spreads, and interacts with human geography, including dialectal variations, linguistic diversity, and the role of language in shaping cultural identity. This field also examines the historical and environmental factors that influence linguistic landscapes, such as migration patterns, colonial histories, and technological advancements.

In recent years, the integration of geolinguistics with advanced technologies has opened new avenues for research and application. Satellite imagery and artificial intelligence offer a unique platform for analyzing linguistic patterns within a geospatial framework. For example, AI-driven models can generate textual descriptions of geographical regions while incorporating linguistic and cultural nuances. These models can aid in understanding how local languages or dialects represent physical and cultural features, such as architectural styles, natural landmarks, or historical sites. This is particularly relevant in multilingual and culturally diverse regions where place names, descriptions, and terminologies reflect deep historical and cultural significance.

One potential application is the use of geolinguistics in automated captioning of satellite imagery. By leveraging geolinguistic insights, AI models can generate more contextually rich and culturally relevant captions. For instance, descriptions of architectural landmarks, such as churches or temples, can include linguistic elements that reflect their historical and cultural context. This approach bridges the gap between visual data and cultural narratives, enabling a deeper understanding of geographical areas.

Geolinguistics also contributes to the preservation of linguistic heritage. By documenting and analyzing the linguistic representations of geographical areas, AI systems can assist in preserving endangered languages and dialects. The integration of geolinguistics into satellite imagery captioning aligns with efforts to map and describe the linguistic diversity of regions, offering tools for researchers, educators, and policymakers. [22]

The interdisciplinary nature of geolinguistics encourages collaboration across fields such as linguistics, geography, cultural studies, and data science. For example, projects like the UNESCO Atlas of the World's Languages in Danger utilize geolinguistic principles to document and protect linguistic diversity globally. Similarly, geospatial AI models enriched with linguistic datasets have the potential to revolutionize how we analyze and interpret cultural landscapes. [21]

Geospatial predictions are also widely used across various domains, including poverty predictions, public health, biodiversity preservation[3], and environmental conservation.

Geolinguistics contributes to the development of search and recommendation systems by enriching textual descriptions with cultural and linguistic context. For example, a search system designed to retrieve satellite imagery based on descriptive text queries benefits from embedding geolinguistic principles into its architecture. When textual queries align with regional linguistic characteristics or cultural references, the search system can deliver more contextually relevant results. This integration creates a semantic bridge between language and location, enhancing both the accuracy and cultural resonance of search results.

1.2 Search engine/Recommendation sytstem

Since this research's applicability is directed toward image-to-text search engines. It should be mentioned that a couple of search variations exist when it comes to image search.

The image-to-image search is also widely known as content-based image retrieval (CBIR). It is a technique used to search for images based on their visual content rather than relying on textual metadata or annotations. In image-to-image search, the user provides an input image, and the system retrieves visually similar images from a database or dataset. There are search engines that support such functionality, such as "Google Images", "TinEye", "Pinterest Visual Search" and much more. Pre-trained convolutional neural networks (CNN) models, such as VGG, ResNet, or EfficientNet, can be fine-tuned on the dataset to extract deep visual features from the images. It is also possible to build an Image-to-Image search tool using CLIP and VectorDBs. [13]

Text-to-image search, also known as reverse image search or content-based image retrieval (CBIR). Unlike traditional image search, where users input images to find similar images, text-to-image search allows users to input textual descriptions or keywords and retrieve relevant images from a database or dataset. Common techniques for feature extraction include word embeddings (e.g., Word2Vec, GloVe) or contextual embeddings (e.g., BERT, RoBERTa).

However, traditional search method mainly relies on keyword input to find matches. For unseen images semantic search is needed.

1.3 Semantic search, comparison

Incorporating large language models (LLMs) into search, enhances user experience by allowing them to ask questions/phrazes and find information more easily. Semantic search is an advanced method of retrieving information that goes beyond traditional keyword-based searches. It aims to understand the context and meaning behind user queries by utilizing text embeddings, which are numerical representations of meaning generated by language models like BERT, GPT, and others. Semantic search improves the accuracy and efficiency of information retrieval.

Semantic search works by transforming both user queries and available documents into numerical embeddings, which represent their semantic meanings. By placing these representations in a multi-dimensional space, the search engine can identify documents closely aligned with the user's query intent, enhancing search accuracy and efficiency. This also explains why vector databases are important for this thesis.[12]

When compared with traditional search, like how browser search engines work, we need to understand that keyword-based search relies on specific words or phrases (keywords) entered by the user to retrieve relevant documents or information. In this approach, the search engine matches the keywords provided by the user with the text content of documents in its index. The search results are typically based on exact matches between the keywords and the text content of documents, without considering the broader context or semantic meaning of the query. While keyword-based search is straightforward and efficient for retrieving documents containing the exact keywords entered by the user, it may lead to limited or irrelevant results if the user's query does not precisely match the content of the documents.

Unlike keyword-based search, semantic search focuses on understanding the meaning and intent behind the user's query, rather than relying solely on exact keyword matches. LLMs can grasp the semantic relationships between words and concepts, handle synonyms and variant spellings effectively, and consider the broader context of the query to provide more accurate and contextually

relevant search results. By understanding the user's query in natural language and analyzing the semantic context, semantic search with LLMs enables more intuitive, personalized, and adaptive search experiences.

There even is a study that compares traditional search and LLM-based search for image geolocation. Which is very on-topic for this research. Participants using LLM issued longer queries, focusing on conversational interactions, while those in the traditional search condition started with shorter queries and gradually expanded them. [14]

Considering LLM's mainly process images and texts to its vectorized forms, so in order to quickly access the information, the vectors must be stored in a vector database.

1.4 Vector databse

Vector databases are crucial for developing LLMs as they enable computers to understand the relationships between words and phrases. In simpler terms, vector databases provide the foundation for LLMs to operate by mapping words and phrases to points in a high-dimensional space. [11]

There are well-known LLMs that utilize vector databases. LLMs, such as GPT-4, understand and generate human-like text proficiently. They convert text into high-dimensional vectors or embeddings that capture the semantic meaning of the text. This enables complex text operations, such as finding similar words, sentences, or documents, making them suitable for chatbots, recommendation engines, and more.

Alternatively, vectors can also be stored in a data frame. Not recommended if the vectors are required to be accessed remotely, however for local testing it can work just as well.

These databases come into play as they are specifically designed to store and manage high-dimensional vector representations efficiently. They can handle the dense, continuous data output produced by language models. For instance, in the context of satellite images, a vector database can be used to store and index vectors generated from images of different geographical locations. When a query is made to find a satellite image similar to a given image, the vector database can efficiently search for the most similar vectors, thereby returning the most relevant satellite images.

The use of a vector database in this context can significantly improve the performance and accuracy of the search system, especially in large-scale and high-dimensional data scenarios. To apply a vector database it is required in this study to Review possible LLM's.

1.5 Large Language Models (LLM) Review

Large Language Models (LLMs) play a pivotal role in generating descriptive and contextually relevant captions for satellite imagery. These models leverage vast amounts of textual data to understand and generate language, enabling them to provide detailed and informative descriptions of visual content. The use of LLMs in satellite imagery captioning enhances the semantic understanding and interpretability of geospatial data.

The concept of Large Language Models (LLMs) traces back to 2017 when Google researchers introduced the Transformer model, a neural network that revolutionized sequential data analysis. This marked the beginning of a new era in language modeling, with the Transformer model gaining popularity for its ability to identify patterns in textual data. Subsequent milestones include the release of BERT (Bidirectional Encoder Representations from Transformers) in 2018, which further advanced language modeling capabilities with its bidirectional approach and large parameter size. In 2020, OpenAI launched GPT-3, a third-generation Generative Pre-trained Transformer with an

unprecedented scale of almost a Trillion words and 175 billion model parameters. This release showcased the immense potential of large language models in handling diverse use cases across industries, from gaming to creative image generation. [4]

Key concepts in the development of LLMs include the use of transformer-based architectures, which have significantly enhanced the modeling of sequential data and language understanding. Architectures like BERT, GPT, and Megatron-Turing have pushed the boundaries of language modeling by incorporating innovative design elements and training techniques. Advancements in LLMs have led to the creation of models with billions and even trillions of parameters, enabling them to handle complex language tasks with remarkable accuracy and efficiency.

The advancements in Large Language Models (LLMs), with their ability to process and generate coherent and contextually relevant textual data, have laid the groundwork for significant innovations in text/image applications. While LLMs excel in handling textual data, their transformative potential has inspired researchers to expand their capabilities into the realm of visual data processing. This intersection of textual and visual modalities has given rise to Vision Language Models (Vision LLMs), which integrate the principles of LLMs with cutting-edge computer vision techniques. These models aim to bridge the gap between text and images, making them particularly suited for complex tasks like satellite imagery captioning, where both linguistic and visual data need to be seamlessly aligned. The following section explores Vision LLMs and their role in leveraging the foundational strengths of LLMs to address text/image challenges.

1.6 Vision LLM Review

Inspired by LLMs, much research started to focus on using large-scale vision-language models for domain adaptions as large vision-language models are a promising way to deal with unseen data. Contrastive language-image pre-training (CLIP) [8], which is the first released pre-trained vision-language model using contrastive learning methods, has been explored by some researchers to handle different classification and segmentation tasks with RS data.

These models, trained on massive amounts of data, have shown impressive performance in various language tasks. While there has been extensive research on language-only models, the development of text/image models that can effectively handle visual input is still evolving. In the context of satellite imagery captioning with a Geolinguistics approach, the use of large language models can be instrumental in generating descriptive and informative captions for satellite images.

We should also mention foundation models. A foundation model is a machine learning or deep learning model that is trained on broad data, making it applicable across a wide range of use cases. These models are general-purpose technologies that can support various applications and tasks. Foundation models can take inputs in a single modality and generate outputs. Watsonx.ai geospatial foundation model – built from NASA's satellite data in collaboration with IBM is the largest of its kind. This model aims to democratize access and application of AI in climate and Earth science, addressing challenges in analyzing vast datasets despite growing data availability. The model, jointly trained by IBM and NASA on Harmonized Landsat Sentinel-2 satellite data, has shown a 15 percent improvement in flood and burn scar mapping over state-of-the-art techniques. It can be fine-tuned for tasks like tracking deforestation, predicting crop yields, or detecting and monitoring greenhouse gasses. [10]

These models can help bridge the gap between visual content and linguistic descriptions, providing a deeper understanding of satellite imagery through textual representations. By utilizing methods such as image captioning and image classification to extract image-as-text representa-

tions, vision LLMs can effectively process visual information and generate informative captions tailored to geolinguistic contexts.

So far for this research, we want to explore the existing models that may provide better or different results. There are a couple of open-source vision LLMs that we can use for testing and analyzing. Such noticeable models such as LLaVA [5], CLIP-rsicd (which is a pretrained CLIP model with Remote Sensing Satellite images and captions)[6], GeoLLM, an applied method that effectively extracts geospatial information from LLMs using auxiliary map data from OpenStreetMap[7].

1.7 Specific Selected Approach

If we want to bridge the gap between visual content and text descriptions, we need to choose an approach that is relevant to the thesis. The first approach would be CLIP which stands for Contrastive Language-Image Pretraining.

CLIP is a model developed by OpenAI. CLIP learns from extensive collections of images and their accompanying text, all without needing to establish explicit connections. Instead, it learns to connect visual and textual representations by emphasizing the match between similar image-text pairs while reducing the match between dissimilar pairs.

CLIP plays a relevant role in the thesis by utilizing the capabilities of large language models and computer vision methods to create a semantic search for the search engine. This is how CLIP effectively connects visual content with textual descriptions by touching upon such concepts like semantic embeddings and zero-shot learning. Whene describing semantic embeddings, CLIP learns to map images and text into a shared semantic space, where similar concepts are represented by nearby points. This enables CLIP to understand the semantic meaning of visual content and textual descriptions and to effectively bridge the semantic gap between the two modalities. CLIP is also capable of zero-shot learning, meaning it can generate accurate descriptions for images without explicit training on paired image-text data. This is particularly relevant for satellite imagery captioning, where obtaining large-scale annotated datasets may be challenging or impractical.

CLIP's ability to understand natural language aligns well with the geolinguistics approach proposed in the thesis. By incorporating geospatial information and linguistic analysis into the training and fine-tuning process, CLIP can generate captions that not only describe the visual content of satellite images but also capture the geographic context and linguistic nuances relevant to the study of geolinguistics.

The training process employs contrastive learning to acquire a unified embedding representation that encompasses both images and their corresponding captions. Within this embedding space, images and their respective captions are brought closer together, as are similar images and similar captions. Conversely, images and captions that do not belong to the same image, or dissimilar images and captions, are likely to be pushed further apart from each other. (Figure 1)

1.8 Transfer Learning and fine tuning

Transfer Learning and Fine-Tuning, though they are closely related and often used together. Transfer learning is a technique that leverages prior knowledge to address the challenge of limited data availability. In standard training processes, models start with randomly initialized weights. Transfer learning, however, provides a more advantageous starting point by utilizing pre-trained weights from a different but related domain. This approach repurposes a model designed for one task to

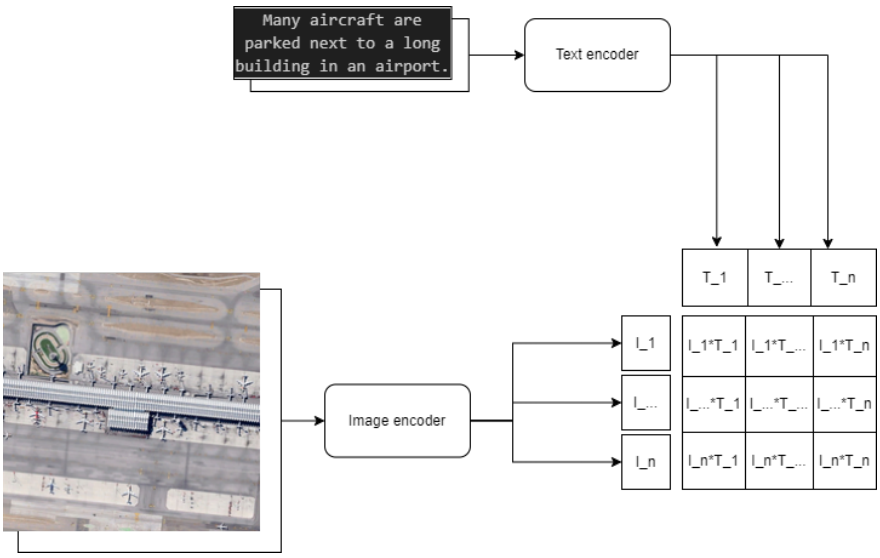

Figure 1. Clip model contrastive pretraining process

perform another, allowing it to benefit from the knowledge learned from a large dataset. As a result, transfer learning improves learning efficiency, especially when working with datasets that have limited samples [16]. Additionally, since it eliminates the need to train a deep learning model from scratch, transfer learning saves significant time and computational resources [17]. However, its effectiveness depends on the similarity between the source and target domain datasets. If the two domains are vastly different, transfer learning may be less effective due to the domain gap.

The goal of transfer learning is to apply the knowledge gained from one task to another. Essentially, it involves transferring the learned information from the layers of one convolutional neural network to another. The knowledge acquired by the original model for predicting certain classes can serve as a foundation for training a new model to predict different, yet related, classes. This approach works best when the domains are closely aligned, with minimal or no domain gap. However, if the domains differ significantly, transfer learning may prove less effective, and training the model from scratch with randomly initialized weights might be a better approach.

Fine-tuning is a widely used and effective method in transfer learning that goes beyond simply leveraging pre-trained weights. It involves customizing a pre-trained model for a specific task and dataset by preserving the features learned from the original dataset while also acquiring new knowledge from the target dataset. This process often includes freezing certain layers of the model—typically all but the last few—so that only the unfrozen layers are adjusted to the new domain. This approach enables the model to retain prior knowledge while adapting to the requirements of the new task.

Fine-tuning approaches can vary depending on the specific task. For instance, in the case of remote sensing imagery, a one-shot fine-tuning method was employed, where the entire Segment Anything Model (SAM) was kept frozen to retain its pre-trained knowledge. Only two parameters were adjusted to optimize performance [18]

Altho, Transfer learning and Fine-Tuning are usually used in tandum, there are ky differences between them and when they are usually used (Table 1.).

Fine-Tuning and Transfer Learning are essential in adapting Vision-Language Models (VLMs), such as CLIP, for specialized tasks like satellite imagery captioning or enhancing semantic search capabilities on specific datasets.

Feature	Transfer Learning	Fine-Tuning
Model Parameters	Pre-trained parameters are typically fixed.	Pre-trained parameters are updated.
Training Requirements	Less computationally intensive.	More computationally intensive.
Flexibility	Limited to using the pre-trained features as-is.	Allows the model to adapt to the new task.
When to Use	When the target dataset is small or unrelated to the source dataset.	When the target dataset is large enough and closely related to the source dataset.

Table 1. Comparison between Transfer Learning and Fine-Tuning

Transfer Learning enables you to harness the general-purpose capabilities of pre-trained models (like CLIP or BLIP) to efficiently handle specialized tasks in satellite imagery. VLMs are pre-trained on large datasets with diverse images and textual descriptions. However, satellite imagery often has domain-specific patterns (e.g., geospatial features, land-use categories) that are not well-represented in general datasets. Transfer Learning allows these pre-trained features to be reused, focusing on satellite-specific adaptations without retraining the model from scratch. Satellite imagery datasets are often smaller and harder to annotate than natural image datasets. Transfer Learning helps leverage pre-trained embeddings, minimizing the need for extensive data.

Fine-Tuning takes Transfer Learning further by adapting pre-trained models to your specific satellite imagery dataset. This is critical for enhancing both captioning and semantic search capabilities. Fine-tuning adjusts the model's weights to learn domain-specific patterns (e.g., recognizing forest boundaries, cloud cover, or man-made structures). This is vital since satellite imagery often involves textures and features different from those found in general datasets. For instance, "urban sprawl near rivers" could be fine-tuned as a valid semantic relationship in your embedding space, improving retrieval accuracy.

1.9 Data preprocessing and feature extraction

Data preprocessing and feature extraction play a vital role in the image search process, enabling the system to effectively analyze image content and align it with corresponding text descriptions [19]. Key steps in this process include:

- Image Resizing: Standardizing image dimensions reduces computational demands for the CLIP model and enhances search accuracy. Techniques like bilinear or bicubic interpolation are commonly used for resizing.
- Color Space Conversion: Transforming images into grayscale or alternative color spaces simplifies analysis and minimizes the influence of color variations on search outcomes.
- Image Normalization: Scaling image pixel values to a consistent range mitigates the effects of differing lighting conditions and contrasts, ensuring consistent search results.

- Feature Extraction: Identifying and extracting meaningful visual features enhances search accuracy. Methods include edge detection, histograms of oriented gradients (HOG), and advanced approaches like convolutional neural networks (CNNs).

The overarching objective of these preprocessing and feature extraction techniques is to prepare images for CLIP model analysis and isolate key visual features for precise matching with textual descriptions. These steps are fundamental to achieving robust accuracy in image search applications.

1.10 Dataset Review

Finding datasets that directly pair satellite imagery with detailed text descriptions is challenging, as most publicly available datasets focus on categorization, segmentation, or detection tasks, rather than providing narrative-style descriptions for each image. The majority of datasets come with labels or annotations that are more suited for supervised learning tasks like classification or object detection, rather than the text descriptions.

When it comes to appropriate datasets, we need to find an open-source dataset that would allow remote sensing capabilities. One of them that we may use is "rsicd". This dataset is designed for the task of creating captions for remote-sensing images. It includes over 10,000 images collected from Google Earth, Baidu Map, MapABC, and Tianditu. The data itself consists of train, test, and valid folders separated for respective use in training and testing the model, filled with photos of satellite images and captions describing the image.

There is also a dataset that is promising but not yet released as of this writing called ChatEarthNet dataset[9]. It aims to facilitate the understanding of complex satellite imagery for common users by providing detailed natural language descriptions alongside the imagery.The dataset comprises 163,488 image-text pairs with captions generated by ChatGPT-3.5 and an additional 10,000 image-text pairs with captions generated by ChatGPT-4vision. The data also goes through manual verification for quality assurance.

On the other hand, we can build our own dataset. For that it is theoretically possible to utilize "Genčių geneologija" from which we take the names, coordinates from Google Maps API, and Wikipedia descriptions based on the names.

1.11 Clip model architecture

The Clip model architecture consists of several components:

1. Input Embedding: The input text is tokenized and embedded into a high-dimensional vector space using pre-trained word embeddings. Let $x_1, x_2, ..., x_n$ be the input tokens, and Embed(x_i) denote the embedding of token x_i.

2. Transformer Encoder: The embedded tokens are passed through multiple layers of Transformer encoder blocks to capture contextual information. Let $h_1, h_2, ..., h_n$ denote the hidden states of the encoder layers.

3. Classification Head: The final hidden state of the Transformer encoder is used for downstream tasks such as text classification. Let h_{final} denote the final hidden state.

The output of the Clip model is obtained by passing h_{final} through a fully connected layer followed by a softmax function to obtain class probabilities.

1.11.1 Training CLIP procedure

The algorithm iteratively optimizes the CLIP model to align images with their corresponding text descriptions. By maximizing similarity for correct image-text pairs and minimizing it for incorrect pairs, the model learns robust text/image representations. These embeddings can then be used for downstream tasks, such as image captioning, zero-shot classification, or image-text retrieval. The code below explains how the model training works from a mathematical perspective (Algorithm 1) [24]. (1) The algorithm trains the model by contrasting correct and incorrect pairs within each batch. (2) The text encoder is adapted to handle text sequences, increasing the model's capacity for complex inputs. (3) Losses are computed and gradients are applied at the batch level, ensuring efficient training on large datasets. (4) The algorithm separates feature extraction, similarity computation, and optimization into distinct stages, making it easy to modify or extend. Note that this is a simplified general training loop, the dataset is also split into train and test datasets.

Algorithm 1. Training Loop for CLIP Model

Number of epochs E, dataset D split into batches $\{B_1, B_2, \ldots, B_k\}$, learning rate η Initialize total loss $L_{\text{total}} \leftarrow 0$ Load and initialize the CLIP model CLIPModel with image and text encoders Set the computational device $\mathcal{D}$ each epoch $e \in \{1, 2, \ldots, E\}$ each batch $B \in D$ Extract Images, Texts $\leftarrow B$ Move images to device: Images $\rightarrow \mathcal{D}$ Initialize batch text features: $\mathbf{F}^t \leftarrow []$ each text sample $T_i \in$ Texts Extract input components $T_i = \{\text{input\_ids}, \text{attention\_mask}, \text{position\_ids}\}$ Compute text features: $\mathbf{f}_i^t \leftarrow \text{TextEncoder}(T_i)$ Append $\mathbf{f}_i^t$ to $\mathbf{F}^t$ Concatenate batch text features: $\mathbf{F}^t \leftarrow [\mathbf{f}_1^t, \mathbf{f}_2^t, \ldots, \mathbf{f}_N^t]$ Compute image features: $\mathbf{F}^i \leftarrow \text{ImageEncoder}(\text{Images})$ Normalize features:

$$\mathbf{F}^i \leftarrow \frac{\mathbf{F}^i}{\|\mathbf{F}^i\|}, \quad \mathbf{F}^t \leftarrow \frac{\mathbf{F}^t}{\|\mathbf{F}^t\|}$$

Compute similarity logits:

$$\mathbf{S}^{\text{image}} \leftarrow \mathbf{F}^i(\mathbf{F}^t)^\top, \quad \mathbf{S}^{\text{text}} \leftarrow \mathbf{F}^t(\mathbf{F}^i)^\top$$

Generate target labels: $\mathbf{y} \leftarrow [0, 1, \ldots, N-1]$ Compute losses:

$$L_i \leftarrow \text{CrossEntropy}(\mathbf{S}^{\text{image}}, \mathbf{y}), \quad L_t \leftarrow \text{CrossEntropy}(\mathbf{S}^{\text{text}}, \mathbf{y})$$

$$L \leftarrow \frac{L_i + L_t}{2}$$

Perform backpropagation:

$$\text{Gradients} \leftarrow \nabla_\theta L$$

Update parameters:

$$\theta \leftarrow \theta - \eta \cdot \text{Gradients}$$

Accumulate loss: $L_{\text{total}} \leftarrow L_{\text{total}} + L$ Compute average loss for epoch:

$$\text{Loss}_{\text{epoch}} \leftarrow \frac{L_{\text{total}}}{k}$$

1.12 Working with long texts

Lets assume we are working with a dataset that utilizes Wikipedia descriptions. Wikipedia descriptions are quite expansive, the original idea is to somehow overcome the fixed token length of 77. By doing so, we need to note that:

- Fine-Tuning: The model may benefit from fine-tuning after changing the context length, as it was originally trained on sequences of length 77.
- Memory Considerations: Increasing the context length will increase memory usage and computational cost during training and inference.
- Interpolation Method: If linear interpolation for positional embeddings doesn't work well, another possibility might be alternative methods like sinusoidal embeddings, which can be more robust for sequence-length scaling.

1.12.1 Increasing CLIP model context length

To train a clip model on a wider text input, just changing the context length parameter is not enough, we also need to:

- Edit the transformer model configuration: In the configuration for the CLIP model, adjust the max position embedding parameter to your desired context length. For example, to double the context length, set this value to 154.
- Adjust the Positional Embeddings: Since the Transformer uses fixed positional embeddings, you'll need to either initialize new positional embeddings with a higher length or interpolate the existing embeddings to fit the new length.
- Adjust Tokenizer Setting: to its new max length parameter. For clip models, they have only one Tokenizer.
- Update the position ids length to match new context length: check position IDs initialization and ensure consistency across model components

One more aspect I want to mention is how CLIP tokenization works. CLIP model tokenizer must process the text before it can be turned into an embedded clip variant. Lets assume the tensor length is 77, and it assigns ID's to parts of the text prompt, referring to specific context (e.g. a word), and pads the text with start and end tokens (usually it's 49406 and 49407 respectively)

While increasing the context length this tensor will expand to 154 (as an example), which is enough to gather the text through the tokenizer. The images themselves do not require increasing the context length, simply passing them through the preprocessor is enough to gather the tokenized version of the image. Note that for the effect to take place the model itself must be initialized with the increased context length, luckily we can do this with the CLIPConfig import from the Hugingface transformers library.

Next, we need to interpolate the positional embeddings into its 3d format. So we reshape the positional embedding weight to use the interpolation and interpolate along the sequence dimension. Now the last dimension is the sequence length. Also replace old positional embeddings with new ones using torch.nograd().

That covers the initialization of the model itself, the tokenizer also needs to be adjusted respectively to accept the the new model's maximum length. After the tokenizer output of the text, we set the positional ids to the maximum length of 154 and arranged them. This will finally allow us to forward pass a single text feature (also considered as the embedded text) with the modified architecture.

A note about the embedded text, it's tensor shape will by default be the length of 512. This is a constant in CLIP considering vit-base-32 model, no matter the context length the fully embedded text will be the length of 512. (e.g. torch.Size([1, 512]))

For token pooling. The neighboring tokens will be averaged, creating new tokens. This usually means that for fine tuning a clip model it will lose purpose of transfer learning.

It is also worth mentioning that There are cases of successful implementations of the clip model with increased context length. A notable CLIP model variation is LONG-CLIP [15]. Long-CLIP introduces an efficient fine-tuning solution on CLIP with two novel strategies designed to maintain the original capabilities, including (1) a knowledge-preserved stretching of positional embedding and (2) a primary component matching of CLIP features.

1.12.2 Reverse approach (Token pooling)

On the other hand, the opposite solution would be to reduce the sequence length for CLIP instead of increasing the context length. Approaches such as Truncating the text inputs themselves are always an option, however, if we are working with a very small dataset, a loss of data will most likely result in poor performance of the finetuned CLIP model.

Other examples of approaches that could be potentially utilized to reduce sequence length for CLIP:

- Average Pooling (Token Pooling):

  You can reduce the length of the sequence by averaging groups of tokens, effectively downsampling by combining the information within neighboring tokens. For example, with a context length of 154, you could average pairs of adjacent tokens to obtain a sequence length closer to 77.

  This approach maintains overall semantics without introducing significant alterations but may result in some loss of fine-grained detail.

- Learnable Downsampling Layer (Attention Pooling):

  Implement an attention mechanism that learns to pool tokens by attending to key token sequences. This approach can capture more information from the original longer context length and condense it into the desired 77-token sequence.

  Attention pooling requires a lightweight trainable layer that can select the most relevant tokens. A simple multi-head attention layer could achieve this.

- Conv1D Downsampling:

  Apply a 1D convolutional layer with stride > 1 to reduce the sequence length. This method effectively merges local token information in a way similar to downsampling in image processing.

By training this layer with the CLIP model, it can learn to aggregate important token information while respecting positional relationships within the original 154-token sequence.

Token pooling in this context is performed using a fixed pooling factor and involves averaging groups of tokens. Let the input sequence of tokens be $\mathbf{T} \in \mathbb{R}^{B \times S \times D}$, where B is the batch size, S is the sequence length, and D is the dimensionality of each token.

- **Input tokens**: $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_S]$, where $\mathbf{t}_i \in \mathbb{R}^D$. - **Pooling factor**: P, the number of tokens to average together. - **Target sequence length**: $S_{\text{target}} = \lfloor S/P \rfloor$.

1. Divide the sequence into non-overlapping chunks of size P:

$$\mathbf{T}_{\text{chunked}} = \{\mathbf{T}[i : i + P] \mid i = 1, P + 1, 2P + 1, \ldots\}.$$

2. Compute the average of tokens in each chunk:

$$\mathbf{t}_j^{\text{pooled}} = \frac{1}{P} \sum_{k=1}^{P} \mathbf{t}_{jP+k},$$

where $j = 1, 2, \ldots, S_{\text{target}}$.

3. Handle remainder tokens ($R = S \mod P$) by truncating the pooled sequence to the desired length:

$$\mathbf{T}_{\text{pooled}} = [\mathbf{t}_1^{\text{pooled}}, \mathbf{t}_2^{\text{pooled}}, \ldots, \mathbf{t}_{S_{\text{target}}}^{\text{pooled}}].$$

Below is the pseudocode representation of the token pooling algorithm:

Algorithm 2. Token Pooling Algorithm

$\mathbf{T} \in \mathbb{R}^{B \times S \times D}$, target length S_{target}. Reduced sequence $\mathbf{T}_{\text{pooled}} \in \mathbb{R}^{B \times S_{\text{target}} \times D}$. Calculate pooling factor $P = \lfloor S/S_{\text{target}} \rfloor$. Initialize pooled sequence $\mathbf{T}_{\text{pooled}} \leftarrow []$. $b \in \{1, \ldots, B\}$ $j \in \{1, \ldots, S_{\text{target}}\}$ $\mathbf{t}_j^{\text{pooled}} \leftarrow \frac{1}{P} \sum_{k=1}^{P} \mathbf{T}[b, jP + k, :]$ Append $\mathbf{t}_j^{\text{pooled}}$ to $\mathbf{T}_{\text{pooled}}$. Return $\mathbf{T}_{\text{pooled}}$.

This algorithm (Algorithm 2) provides an efficient mechanism to reduce sequence length in LLM models, improving computational performance. To refer the original paper: Efficient Transformers with Dynamic Token Pooling [20].

Note: As far of this research there is also another way to increase the context length. The third option being attention pooling, (like in Vision Transformers). This would allow the model to learn where to pay attention when reducing from 154 to 77 tokens. To fit into the scope of this research we will not test the attention pooling strategy.

2 Methodology

This is the proximate flowchart of experiments that will be used to test the models (Figure 2). This section describes further details.

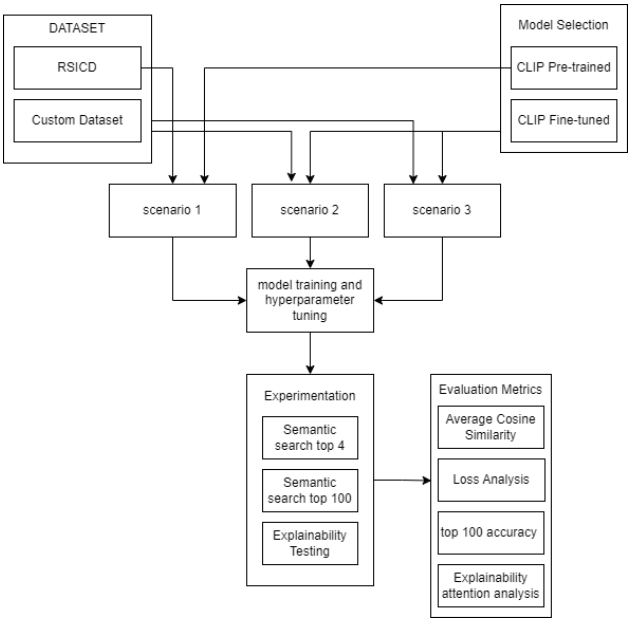

Figure 2. experiment flowchart

2.1 Scope

The scope will be split into a couple of main steps.

- Step 1. Take a data set with photos and descriptions (publicly available). Photo ID and text description. Make text, query embedding. Use co-similarity to find the best match between the user's embedding and the description of the query. After finding the image ID to return. Search Engine. Use a model to build a text embedding (vector DB).
- Step 2. From geoportal.lt take an orthographic photo. Take the model, vision-languege and test a description for the orthographic photo. Test the search system implemented in the first step with the new photo and a different text. Feed the text into the model and make a prediction, and the output is the closest photo.
- Step 3. Data collection. Gather Wikipedia descriptions of all Lithuania's churches. For this, we will use the dataset that was shared by "Genčių genealogija". We will use this as a list to gather Lithuanian church descriptions, and by the geo-location point take the orthographic photos. Thus we will construct the custom dataset. The resason to specify only on Lithuania's churches is to specify the dataset to a more local environment and test out if a small dataset will suffice as a CLIP model dataset.
- Step 4. Train the clip model on the new dataset (Specifically fine-tuning) and conduct further experiments, such as cosimilarity rates, amount testing, loss curve performance, attention vizualization.

2.2 Data Collection

Primarily for testing purposes any dataset will suffice, at first it was decided to use the rsicd dataset. This dataset is designed for the task of creating captions for remote-sensing images. Since our main objective is for the specific use of a search system, the dataset can be potentially applicable to search.

2.2.1 Description comparison dataset

The rsicd dataset includes over 10,000 images collected from Google Earth, Baidu Map, MapABC, and Tianditu. The data itself consists of train, test, and valid folders separated for respective use in training and testing the model, filled with photos of satellite images and captions describing the image. We may use this dataset as a comparison for our custom dataset.

Link to dataset: https://paperswithcode.com/dataset/rsicd

2.2.2 Custom dataset construction (text description)

More time was required to develop a custom dataset with text-image pairs that have a more descriptive Wikipedia-style caption. Considering There are approximately 776 churches in Lithuania and there was no way to effectively automate the description gathering, It had to be done by hand.

Reasons for the limited automatic gathering of description:

- Title inconsistency. Even though the Wikipedia API does exist, to gather the page contents you must provide the exact title. The list of churches that was provided to me had a lot of titles that were slightly different, the usual example being switched words (e.g. "Šventoji Pauliaus...", "Pauliaus Šventoji...").
- Content verbosity. The descriptions had too much information that was not relevant to the dataset. The segments were also too long to train the model effectively. Thus it was required to summarize and be very selective in the descriptions that were collected
- Inconsistent page structure. For this model, the training data from the descriptions should be very visual, which is why it was required to take the architectural descriptions of the churches. The issue arises when the Wikipedia page does not have an architecture section, meaning the architecture description was not provided or it was included in the main description or the story of the church. Which is why I had to gather the architectural description by hand.
- Missing Wikipedia page. There were a few churches that did not have a Wikipedia page, being either old and abandoned or converted into storage facilities after the Second World War. However, there is some information about these churches on the local living area websites. Hence 776 churches is not a lot of data (In other words every single line is valuable in this case), I wanted to make sure I wouldn't assign such churches as outliers.

The process of description gathering itself took approximately two full weeks.

Hence I was required to gather/create my own descriptions, I had to create guidelines by which I deemed the text description viable for clip model training.

Description gathering guidelines:

- Because we are training a CLIP model with satellite images, the descriptions themselves need to be about the church's exterior view.
- The length of the text needs to be small enough to train the model.
- Do not include history or other factual information that does not describe the architecture of the church or its close environment by itself, that includes names of builders, priests and memorials with description plates.

2.2.3 Custom dataset construction (Satelite images)

For satellite images it was especially hard hence I primarily wanted access to geoportal images, however they do not have the API that would allow me to automate the gathering process, and considering the time limitations It was not possible to train the model immediately. So I waited for a response to gain access to individual photos of satellite images from one of the University members, however, I have not gotten a response from them or access to satellite images. The last option I decided to move forward with is by using Google Maps API to gather satellite images based on the geo-points provided by "Genčių genealogija" from hnit-baltic. [23].

link to the dataset hnit-baltic:
https://www.arcgis.com/home/item.html?id=feb35d29fa9f4582a71d728c77ef286e

First of all, I needed to convert the coordinate system of the provided list from LKS-94 to WGS-84 for the latitude and longitude to use GoogleMaps API. Second I created an API key to allow myself to access the API service. And lastly, the image size taken of the areas was 600x400 px.

All images were saved as PNG formats for further use in training the CLIP artificial intelligence model.

2.2.4 ChatGPT Description dataset

Since the custom dataset will be constructed manually, it is important to ensure that we also include a control dataset to provide a standardized baseline for comparison. To achieve this, we will leverage ChatGPT to generate high-quality descriptions for the control dataset. These descriptions will be carefully tailored to align with the specific requirements of our CLIP model training, ensuring consistency and relevance across the dataset. This approach will not only save time but also enhance the overall quality and diversity of the data used in training. For the caption provided to ChatGPT was "read through the name column of this csv. These are the names of all Lithuanian churches, please generate descriptive captions for each church. The descriptions must be in English, try to avoid church names in the description. The descriptions must talk about the church's exterior architecture." The reason for such a prompt was that the model when trained would not reference the church picture by name, but by it's exterior architecture. Since I did not have a payed version of chat GPT, the entire csv file of church names needed to be copy pasted into chat GPT and copy segment by segment of small portions of church descriptions into the new dataset. The result of this data collection is a dataset of google satellite images and chatGPT Descriptions, resembling similar in length to the manually created dataset.

2.3 Model Selection

The model Selected will be CLIP. It learns to connect visual and textual representations by emphasizing the match between similar image-text pairs while reducing the match between dissimilar pairs. It will be a pre-trained model at the start, and the trained finefuned model from our custom dataset later on. There is also a clip-rsicd-v2 clip model from hugingface repository that was tested to see the difference between the basic ViT-B/32 clip model for semantic search capabilities.

We will use this model for image and text embeddings and use co-similarity to find the best match between the user's embedding and the description of the query. The similarity score will then be used to display the top images that match the user query embed.

2.4 Technical challenge and approaches

The technical challenge for this thesis is overcoming the context length. From the literature review it was selected to test how token pooling compares with increased context length. Approaches such as Truncating the text inputs themselves are always an option, however, we are working with a very small dataset of 775 image-text pairs, thus a loss of data will most likely result in poor performance of the finetuned CLIP model. The number 154 will be common as it is the longest text description in the dataset. For token pooling the tokenizer will increase the token length to 154 in order to fit all descriptions and after that the tokens will be pulled together to create a context length of 77. That means two neighboring tokens will get averaged. And the other scenario would be simply creating a configuration for the clip model that increases the context length up to 154 in order to fit the longest description.

2.5 Evaluation Metrics

For evaluation, our main metric will be the cosine similarity score.

Cosine similarity between vectors $\mathbf{a}$ (the embedded text) and $\mathbf{b}$ (the embedded image) is calculated as:

$$\begin{aligned}\text{Cosine Similarity}(\mathbf{a}, \mathbf{b}) &= \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \\ &= \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}\end{aligned}$$

We will also look into CLIP explainability to analyze how the model perceives and directs its attention to the photo when given the appropriate text prompt. The output should be a heatmap.

The loss function used in the CLIP model training combines contrastive loss for both image-text pairs. This will help us see the change of loss when fine-tuning the clip model. The general loss function for CLIP is:

$$\mathcal{L} = \frac{1}{2}\left(\mathcal{L}_{\text{image}} + \mathcal{L}_{\text{text}}\right)$$

where:

$$\mathcal{L}_{\text{image}} = \frac{1}{N}\sum_{i=1}^{N} -\log \frac{\exp\left(\text{sim}(x_i, y_i)/\tau\right)}{\sum_{j=1}^{N} \exp\left(\text{sim}(x_i, y_j)/\tau\right)}$$

$$\mathcal{L}_{\text{text}} = \frac{1}{N} \sum_{i=1}^{N} -\log \frac{\exp\left(\text{sim}(x_i, y_i)/\tau\right)}{\sum_{j=1}^{N} \exp\left(\text{sim}(x_j, y_i)/\tau\right)}$$

Average similarity will also be taken into account. After computing the cosine similarities for all pairs, the results are flattened into a scalar list:

$$\text{cosim\_list} = [\text{cosine\_similarity}(\mathbf{t}_i, \mathbf{v}_i) \; \forall \; i]$$

The flattened list is then sorted in descending order:

$$\text{sorted\_cosim\_list} = \text{sorted}(\text{cosim\_list}, \; \text{descending})$$

The average cosine similarity of the sorted list is computed as:

$$\text{average\_cosim} = \frac{1}{N} \sum_{i=1}^{N} \text{sorted\_cosim\_list}[i]$$

Where N is the total number of elements in the list.

And lastly, there will be Amount Testing. This means that we will print the top 100 most similar images and manually check how correct the prompt is to the image. I determine if the correct images have a specific word or words in their resembling texts. For example if the prompt is "The church has a classical, neoclassical, or classicist design." I check the corresponding images that are associated with the text that contains the word "classical", if the prompt is "The church is an example of Baroque architecture." I check the corresponding images that are associated with the text that contains the word "Baroque". For the query "The church is rectangular in plan, with two towers." and models that utalize the RSICD, I validate manually, looking if the top down design of the church is rectangular and if the church has two towers. No labeling was applied, mostly this test was done manually.

2.6 Hyperparameter fine-tuning

The parameters might be somewhat better if we apply parameter optimizations.

What was done to optimize fine-tuning, (Note this is the list of used optimizations to test the loss, not neceserally something kept after optimizations):

1. **data split**
    - Data split was 80/20 percent split. This is commonly known as the best train and test split for the dataset.
    - Fundamental to the development of robust and reliable machine learning models. Put very simply: The goal: Create a neural network that generalizes well to new data.
2. **Hyperparameter optimization**
    - For hyperparameter optimization Ray tune was used. Ray tune is a library for hyperparameter tuning and experiment management. It is part of the larger Ray ecosystem, which provides a distributed computing framework. Ray Tune allows you to efficiently and easily optimize machine learning models by automating the process of searching for the best hyperparameters.

- For testing out different hyper parameters, such parameters where used and altered:
  - "lr": Learning rate - determines the size of the steps an algorithm takes while optimizing the model's parameters (like weights and biases) during gradient descent. perhaps the most important hyperparameter
  - "batch_size": batch size - defines the number of training examples processed in one forward/backward pass before updating the model's weights.
  - "weight_decay": Weight Decay - a regularization technique used to prevent overfitting in machine learning and deep learning models. It works by penalizing large weights in the loss function, encouraging the model to keep its weights small.
  - "epochs": Epochs refer to the number of complete passes through the entire training dataset during the training process
  - "dropout": is a regularization technique used in neural networks to prevent overfitting. It works by randomly "dropping out" (i.e., setting to zero) a percentage of the neurons during each forward pass of training. This forces the network to be more robust and prevents it from relying too much on specific neurons.
  - "warmup_epochs": Warmup Epochs refer to the initial training period where the learning rate is gradually increased from a small value to the target (or base) learning rate.
- to run these tests an environment with a powerful GPU was used, "Google Colabs". To run through testing trials a T4 GPU was used with increased RAM. A100 was not necessary because it requires a lot of computing units for which I had a limited amount of.

3. Warm-up learning: This is done to stabilize training, avoid large weight updates at the start, and help models, especially large models like transformers, converge more smoothly. When training deep neural networks, especially with optimizers like Adam or SGD with momentum, large updates at the beginning can destabilize training. This is especially true for models like transformers, BERT, ResNets, CLIP etc.

4. Transfer learning: Using another model's weights as a starting point to train our dataset. Essentially this is finetuning by itself. Very useful since we have a small dataset. As a base for transfer learning the vit-B32 model was tested and flax-community/clip-rsicd-v2 was tested. Utalized clip-rsicd-v2 and froze the image layers to hence it is better fitted for our dataset.

5. dropout layers: to prevent overfitting. It works by randomly "dropping out" (i.e., setting to zero) a percentage of the neurons during each forward pass of training.

6. early stopping was used to prevent the process running if the loss has not decreased over certain epoch. This can be handled by Ray Tune.

7. evaluation metrics: Loss, Val_Loss (Validation Loss), Top1 (top 1 accurasy), Top 5 (top 5 accurasy)

3 Experiments

3.1 Configurations and experimental design

For experimentation, we will first use the rsicd dataset at first, then conduct expiriments with the custom church dataset with our trained clip model.

- Step one will be to create a semantic search and test it, it should display the top 4 images similar to the given text prompt. The model then will be fine-tuned and be tested on the Loss performance. After which a heatmap will be displayed of randomly chosen image text pairs from a cosimilarity matrix. The Average cosimilarity of the entire dataset pairs will be provided.
- Second will be the Amount Testing, printing the top 100 most similar images and manually checking how correct the prompt is to the image. Since there are quite a lot of images this paper will not include all of them.
- Lastly, there will be CLIP explainability tests, showing attention heatmap for the given image and prompt. This will help with the understanding of how the model is viewing the photo and directing its attention towards.

The scenarios for these tests are Clip models finetuned with difference in architecture or tweeks in tokinization. Here is the table displaying the scenarios for this test. (Table 2)

scenario	**experiment description**
CLIP-opensource pre-trained	CLIP Model pre-trained with an open source text/image satellite imagery, to validate the functionality of the CLIP model as a baseline.
CLIP-longcontext	CLIP Model fine-tuned with an extended token context length on top of clip-vit-base-patch32
CLIP-tokenpooling	CLIP Model fine-tuned with tokens averaged together on top of clip-vit-base-patch32

Table 2. Table of scenarios

3.2 Results

3.2.1 CLIP-opensource pre-trained

Step 1: similarity rate test

The result of the semantic search given the query "Highway" (Figure 3-a). The similarity rate reaches 28 percent, which seems this is enough of a similarity percentage to detect that there is indeed a highway or a form of a road in an image. The reason why we see a seemingly low.

The result of the semantic search given the query "Highway" (Figure 3-b). The similarity rate reaches 26 percent, which is unexpectedly slightly lower or similar to the default ViT-B/32 clip model.

If we take a longer query such as "Many aircraft are parked next to a long building in an airport" (Figure 4-a) we can see that there are indeed airports or lots of aircraft visible in the photo as well

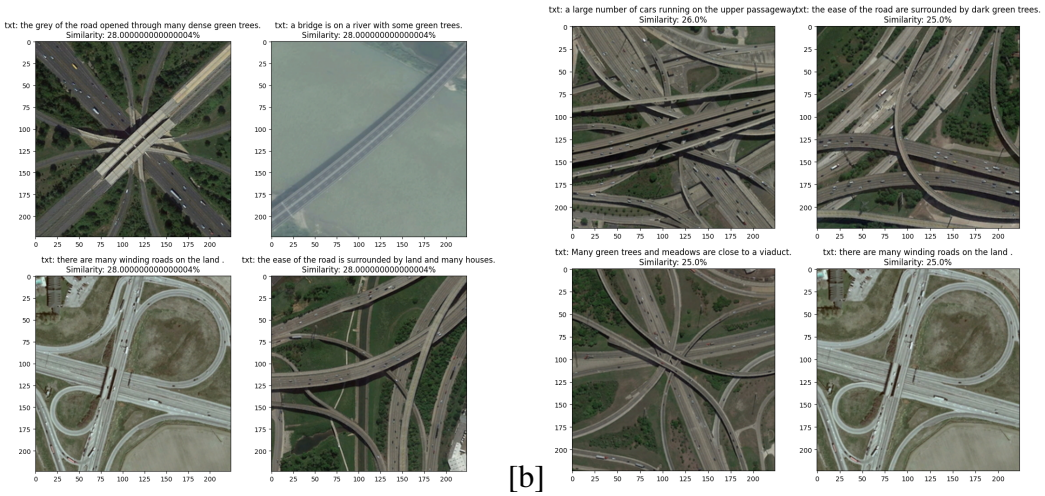

Figure 3. (a) result of the semantic search given the query "Highway" clip-vit-base-patch32, (b) result of the semantic search given the query "Highway" with a pre-trained clip-rsicd-v2 CLIP model

as a long building. The similarity rate also reaches up to 36 percent between the query and image. So in this case a more detailed description might mean the model is referencing a larger area in the photos. This will be clearer when Testing CLIP explainability.

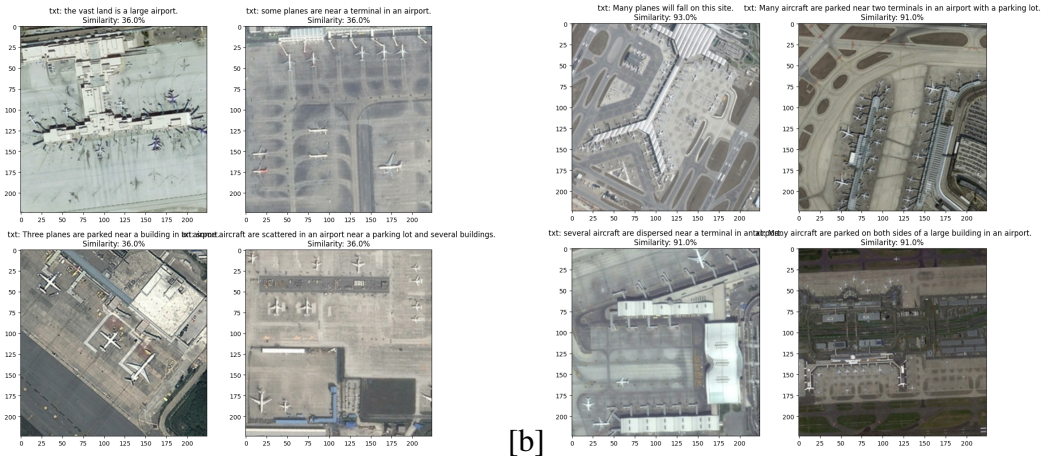

Figure 4. (a) The top 4 images based on query "Many aircraft are parked next to a long building in an airport" (b) results for image-image search

Figure 5. example photo of an airport

By doing different search approaches, in this case, Image-to-image We can expect the clip

model to output similar images at a noticeably higher similarity rate in this case up to 93 percent. The comparison works the same way, we embed the images, which turns them into tensors a synonym for a vector, and creates Cosine similarity between two vectors, the smaller the angle the more similar the images are. (figure 5) (figure 4-b)

This test was not particularly necessary for the thesis, but it allows us to understand that in the semantic search similarity between two image vectors is higher than image and text.

For the open-sourced clip pretrained model it was tried to replicate the training process, according to the documentation for finetuning they used a batch size of 32. The best training results were observed using the Adafactor and Adam optimizers with a learning rate of 5e-5 and a linear learning rate schedule of a factor 20%. In my Implementation according to CLIP RSICD I had to finetune the model as it did not provide the same results. So in my case I had to increase the warmup ratio to 0.4 (or 40%) and learning rate to 1e-6 and reduce the batch size to 16. (Figure 6) If we take the average Cosine Similarity of the training data the result will be: 68.5%

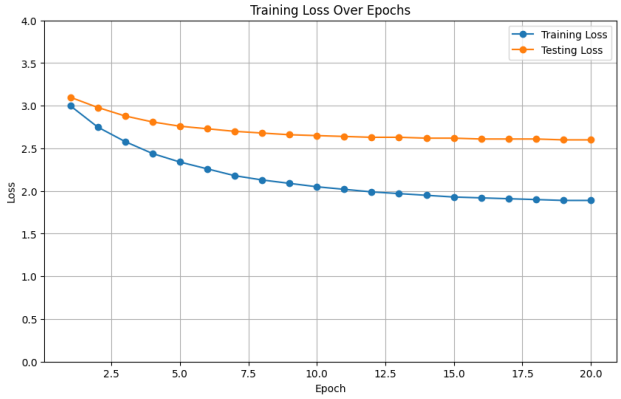

Figure 6. Attempted implementation of the finetuned CLIP-rsicd

Step 2: Amount Testing

The 100 pictures searched had a lot of large highway intersections, however, it technically is a highway and will not be counted as an incorrect image. In the figure (Figure 7) it is displayed the incorrect images that did not match the query "Highway going through a forest". The criteria is that there is supposed to be a visible highway/road (highway through interceptions is also allowed) and a visible forest next to a road. The outliers either did not have a forest visible, or instead of the forest there might have been a few placed trees alongside the road, that do not classify as a forest. Sometimes polluted rivers could be confused with roads. And there were a lot of images that had a forest, however, no visible road was seen. In total 9 out of 100 incorrect images were described.

This manual testing was conducted on multiple queries (Table 3). The manual testing efficiency can be seen as a percentage. Another noticeable observation is that because of the small dataset, there is less specific data to sort first, so it's possible that these outliers might occur because of the lack of a bigger dataset.

Step 3: CLIP explainability testing

Let's test out what will happen if we utilize the geoportal images of known landmarks (for now let's limit ourselves to the locations of churches) and compare Wikipedia descriptions of those

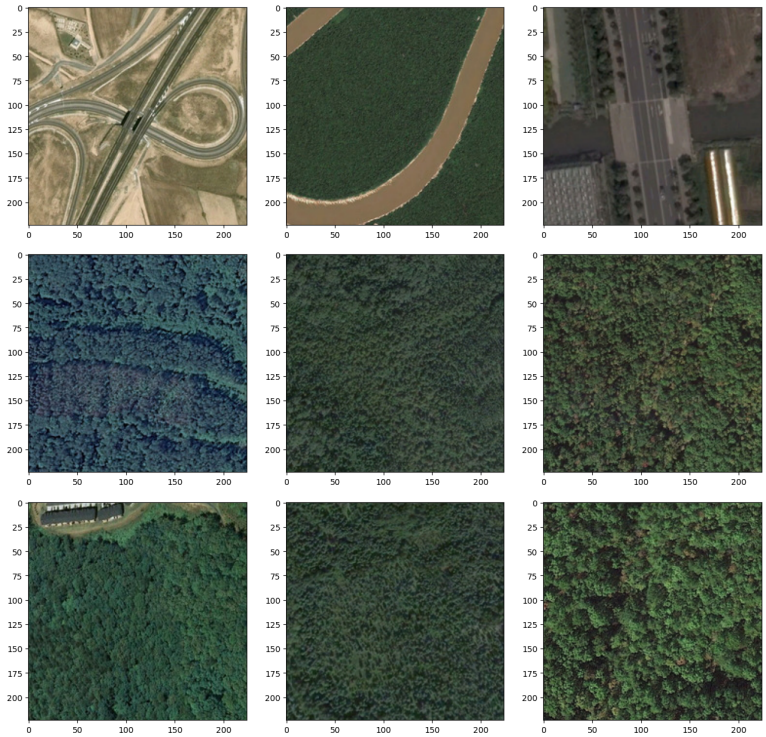

Figure 7. selected outliers of "Highway going through a forest"

query	total image count	incorrect images	accuracy percentage
Highway going throw a forest	100	9	91
a church	100	31	69
intersection with trees	100	23	77

Table 3. table of queries experiments on the pre-trained model

locations. The query for this test is "Cathedral Basilica of apostles St. Peter and St. Paul of Kaunas. The cathedral, being 84 m long, 28 m high, and 34 m wide, is the largest Gothic church in Lithuania. The Chapel of the Blessed Sacrament, built in 1895, is an independent extension of the southern nave with carved wood furnishings". Querry had to be chopped off because of the 77-word limit, or context length equal to 77. This persists through ViT-B/32, ViT-B/16, or ViT-L/14 CLIP versions. The similarity rate for the base ViT-B/32 CLIP model is 20.3 percent. This is not abnormally horrible, however, fine-tuning it to the dataset with Wikipedia captions and images of (for example churches) will be important. We can see how well this image was read by the model, its attention focuses on the church primarily. (figure 8)

Here we test out if a Lithuanian text is understandable to the pretrained model (figure 9). As it is observed the model cannot understand the given caption. So its attention is not focused. Even tho its cosine similarity rate is at 23 percent, (3 percent higher than English) it just shows that the similarity rate is not the only metric for captioning. Caption is "Kauno šv. apaštalų Petro ir Povilo arkikatedra bazilika bažnyčia Lietuvoje, stovinti Kaune, Senamiestyje, prie Rotušės aikštės".

This tests out simpler, non-Wikipedia descriptions of the attention capabilities of the model (figure 10). Here we use an image from the existing dataset with the caption of "Many aircraft are parked next to a long building in an airport." We can see that the attention is more directed toward the line of aircraft, heavily emphasizing the word aircraft.

By the table below (Table 4) we can tell that the model attention works the worst with Lithua-

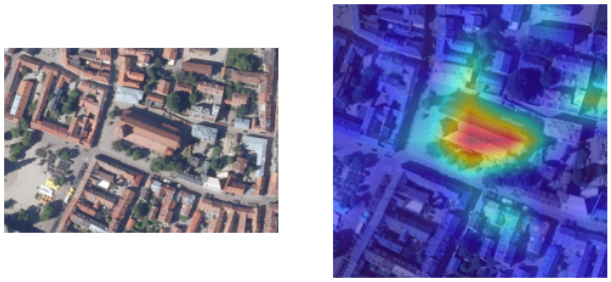

Figure 8. Cathedral Basilica of apostles St. Peter and St. Paul of Kaunas. With en wiki text captioning on pretrained models

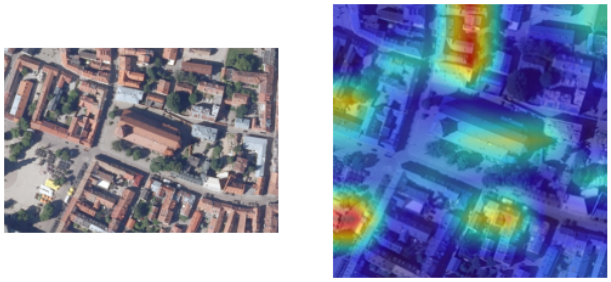

Figure 9. Lithuanian caption attention test

nian text.

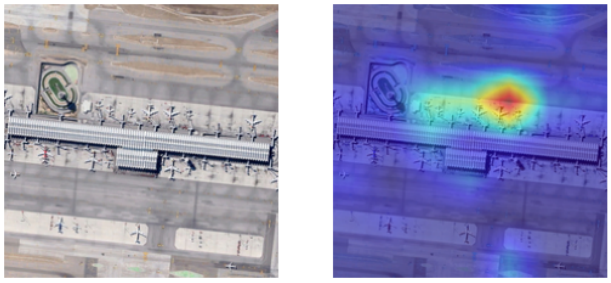

Figure 10. aircraft heatmap

query	attention	image
Cathedral Basilica of apostles St. Peter and St. Paul of Kaunas. The cathedral, being 84 m long, 28 m high, and 34 m wide, is the largest Gothic church in Lithuania. The Chapel of the Blessed Sacrament, built in 1895, is an independent extension of the southern nave with carved wood furnishings"	centered	Kaunas Church
Kauno šv. apaštalų Petro ir Povilo arkikatedra bazilika bažnyčia Lietuvoje, stovinti Kaune, Senamiestyje, prie Rotušės aikštės	dispersed	Kaunas Church
Many aircraft are parked next to a long building in an airport	centered	Airport

Table 4. attention heatmap experiments

3.2.2 CLIP-tokenpooling

For the next experiment we will test the top images and similarity scores of our custom dataset fine tuned model. The model was trained with 20 epochs without hyperparameter optimization. These images (Figure 11) were gathered from the trained model using token pooling to reduce the token length for training. The prompt caption given for this retrieval task given was "The church is classical, rectangular in plan, with two towers.", a prompt given for specifically churches. In the images (Figure 11) the similarity score is low, reaching up to 25 percent. The feature of classical architecture was not noticed based of the satellite images, the context of "rectangular plan" can be noticed in all images, the context segment of "with two towers" has been noticed in the first three out of four images.

Figure 11. Top 4 images of the caption "The church is classical, rectangular in plan, with two towers." Model trained with token pooling

If we look at images in (Figure 12.) and (Figure 13.) the results are different this time. For this the token length was pooled and the chatGPT dataset was used. The similarity rate is low (24.21 and 24.29 percent) but not that different from the tests with custom dataset counterpart (Figure 11). We may also note that the top image of (Figure 12) church was not rectangular but spherical, so we can say that the model did not follow the description provided in the query.

Let's try to optimize the model. If we apply hard negative mining. Weighting hard negatives forces the model to separate difficult examples, improving the representation quality. Theoretically it should lower the loss curve at least by some amont. Note that we can not use very sharp temperatures for the loss, as it will plateau immediately, in this case the temperature was not applied, just the margin. What we will also apply is gradual unfreezing. Essentially freezing the previous layers at the start, then gradually unfreezing later on in the epoch run. Specifically the vision parameter will start unfreezing at the 6-th epoch by 5 layers and the text model parameters will be unfrozen at 10-th iteration. Over time, starting with the layers closer to the output and moving towards the

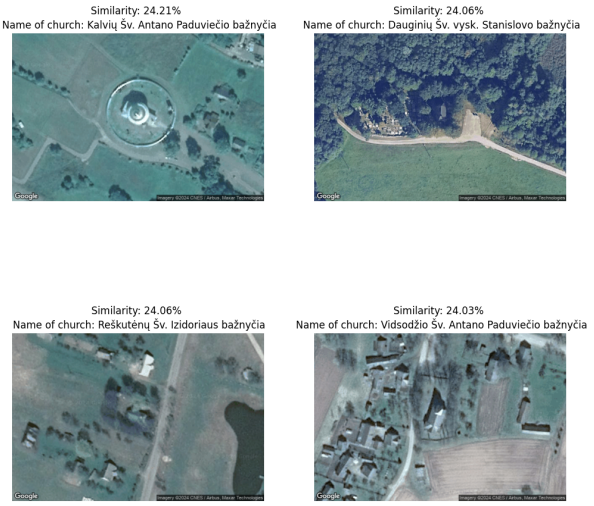

Figure 12. Top 4 images of the caption "The church is classical, rectangular in plan, with two towers." Model trained with token pooling and ChatGPT dataset

Figure 13. Top 4 images of the much longer caption "The new brick church stands without the planned 33-meter high bell tower. A small, metal-framed bell tower has been erected in the churchyard..." Model trained with token pooling and ChatGPT dataset

input layers. This allows the model to adapt to the new data while retaining the general knowledge from pre-training. Since the images are different to the pretraining model we allow the model to unfreeze them first, afterwards incorporating texts, unfreezing both at the same time might cause important data to be forgotten (Note for token pooling strategy this will most likely not have an effect but testing might be useful). To fine-tune the parameters we will use Ray tune. Here are the parameters decided to run through and reasoning why they were chosen:

Learning Rate (lr) (5e-6, 3e-4) Controls the step size for weight updates. Small values ensure stable fine-tuning for pre-trained models. Batch Size ([32, 64]) Number of samples per batch. Larger batch sizes improve negative sampling but require more GPU memory. Weight Decay (1e-5, 1e-2) Regularization technique to prevent overfitting by penalizing large weights. Number of Epochs ([20, 30]) Number of complete passes through the dataset during training. Dropout Probability (0.1, 0.3) Probability of dropping neurons during training for regularization. Helps avoid overfitting. Warmup Epochs ([3, 5]) Number of epochs with gradual learning rate increase to stabilize training. Margin ([0.05, 0.1, 0.2, 0.3, 0.5]) Minimum separation distance between positive pairs and negatives in contrastive loss.

After fine-tuning our result was this table(Table 5). Note that this table was truncated in size to neatly fit into the documentation, the actual ran tests where abut 25 with greace period of 5 epochs essentially the fine tuning process took approximately 5 hours. The table presents the results of hyperparameter tuning for the CLIP fine-tuning process using various configurations. Each trial was evaluated across key hyperparameters, including learning rate (lr), batch size, weight decay, number of epochs, dropout rate, warmup epochs, and margin. The performance of each trial was assessed using training loss, validation loss. Top-1 accuracy, and Top-5 accuracy are not necessary for such dataset.

One notable trend is that configurations with higher learning rates (e.g., 0.000163302 and 0.000274611) and moderate weight decay (e.g., 0.000667683) tended to achieve lower validation loss. This suggests that slightly larger learning rates allow the model to explore the parameter

space more effectively, especially when combined with 5 warmup epochs, which give the model time to stabilize before applying full learning rate updates.

Smaller batch sizes (32) were often associated with lower validation loss and better Top-1 and Top-5 accuracy. This is likely due to the model having more gradient updates per epoch, leading to more frequent weight updates. Trials with larger batch sizes (64) generally exhibited higher validation loss, suggesting that fewer updates per epoch may have limited the model's ability to generalize. Trials with lower dropout values (e.g., 0.128557 and 0.104004) performed worse, indicating that regularization through dropout is essential to avoid overfitting. Trials with higher dropout values (e.g., 0.2059 and 0.2846) achieved better generalization.

The use of the margin parameter significantly influenced performance. A margin of 0.2 was the most common across trials, and it appears to strike a good balance between pushing positive and negative pairs apart. Trials with smaller margins (0.05 and 0.1) generally had worse Top-1 and Top-5 accuracy, possibly because they were too lenient on hard negatives. Larger margins (e.g., 0.5) also showed poorer results, as they likely made optimization more challenging.

In summary, the most successful configuration achieved a validation loss of 1.9. This configuration used a learning rate of 0.000163302, batch size of 32, weight decay of 0.000667683, 20 epochs, a dropout rate of 0.2059, 5 warmup epochs, and a margin of 0.2. This combination provided an effective balance of exploration (learning rate), regularization (dropout, weight decay), and generalization (margin).

LR	Batch Size	Weight Decay	Epochs	Dropout	Warmup Epochs	Margin	Iter	Total Time (s)	Loss	Val Loss	Top-1 Acc (%)	Top-5 Acc (%)
4.62e-5	64	2.07e-4	20	0.2846	3	0.2	20	1363.39	3.9443	3.6659	12.77	52.48
1.63e-4	32	6.68e-4	20	0.2059	5	0.2	20	1336.94	2.8125	1.9314	4.00	20.00
4.99e-5	64	1.32e-3	20	0.2565	3	0.2	5	267.51	4.1249	3.8469	1.42	4.26
2.75e-4	32	5.82e-3	20	0.2106	5	0.2	20	1340.22	3.3277	2.5079	1.33	6.67
2.88e-5	64	8.05e-5	20	0.2357	5	0.2	5	264.84	4.1260	3.8483	0.71	3.55
7.58e-5	32	6.40e-5	20	0.1286	5	0.5	10	511.57	3.4168	2.6981	1.33	6.67
9.59e-5	32	2.89e-4	20	0.2083	5	0.1	20	1345.53	3.2318	2.2882	6.67	33.33
6.48e-6	32	2.16e-4	20	0.2225	5	0.5	5	259.15	3.7907	2.9675	2.67	13.33
5.17e-6	32	2.06e-3	20	0.2914	3	0.2	10	514.89	3.4312	2.7100	1.33	6.67
9.01e-5	32	1.74e-3	20	0.1746	3	0.5	10	514.45	3.4160	2.6913	6.67	33.33
2.02e-5	32	1.54e-4	20	0.2090	5	0.1	10	512.34	3.0193	2.3059	1.33	6.67
6.48e-6	64	6.35e-5	20	0.2110	3	0.5	5	268.33	4.6654	4.3885	0.00	1.42
5.45e-6	32	2.62e-3	20	0.2776	3	0.5	5	261.61	3.8434	3.0227	1.33	6.67
...												

Table 5. Hyperparameter Tuning Results for Fine-Tuning CLIP with token pooling

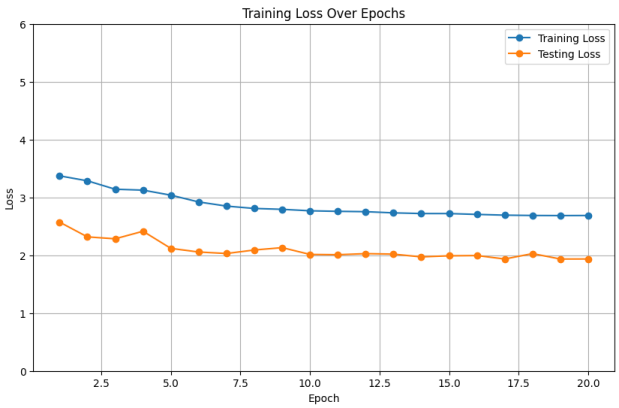

Figure 14. loss after optimization

The model in Figure 14 (after optimization) is more optimized. The hyper-parameter tuning process has resulted in: Steady and meaningful reductions in both training and testing loss. Improved generalization, as evidenced by the smaller gap between training and testing loss. A lower

overall testing loss, which is critical for real-world performance. The configuration used in Figure 14 should be preferred, as it achieves better learning and generalization, making the CLIP model more suitable for deployment or further fine-tuning.

After the tests, it seems that the optimized model has an average cosine similarity of 61 percent.

Figure 15. Top 4 images of the caption "The church is classical, rectangular in plan, with two towers". Model trained with token pooling after hyper parameter tuning

Figure 16. Top 4 images of the much longer caption "The new brick church stands without the planned 33-meter high bell tower. A small, metal-framed bell tower has been erected in the churchyard...". Model trained with token pooling after hyper parameter tuning

As we can see in both images (Figure 15 and 16) the similarity for short captions is about 50 percent and for the long caption is about 47 percent. Much higher than with the non-fine tuned parameters. We can see the results of square plan in all images, however some images repeat, indicating that some co similarity confusion might still happen.

The provided heatmap (Figure 17) represents the cosine similarity between text descriptions of churches and their corresponding image features after applying the fine-tuned clip model in figure 14. The diagonal elements have the highest similarity values (close to 1), indicating that the model correctly aligns most text descriptions with their corresponding images. This is expected, as the model is trained to maximize similarity between matching text and image pairs. Off-diagonal elements generally have lower similarity values, which is a positive sign as it indicates the model distinguishes between unrelated text-image pairs. However, there are some off-diagonal elements with moderate values (e.g., 0.66, 0.74), which might indicate slight confusion between some descriptions and images. The heatmap demonstrates that the CLIP model aligns text and image pairs effectively, with a strong focus on matching pairs (high diagonal values). However, there is room for improvement in handling subtle distinctions between similar text descriptions, which could be achieved through additional data augmentation or fine-tuning with harder negatives.

When using a different data set, a much larger open source dataset just like RSICD, that was used in the first scenario, and train it on our Clip model that utilizes token pooling, the resulting Loss graph will look like this (Figure 18). The parameters where learning rate of 5e-05, weight decay 0.1, batch size 32 and used AdamW optimizer. The loss curve demonstrates that token pooling effectively enables the CLIP model to learn meaningful representations for the RSICD dataset, achieving both convergence and good generalization. The alignment of the losses reflects a well-tuned model with minimal overfitting. In contrast, the custom dataset preforms worse (Figure

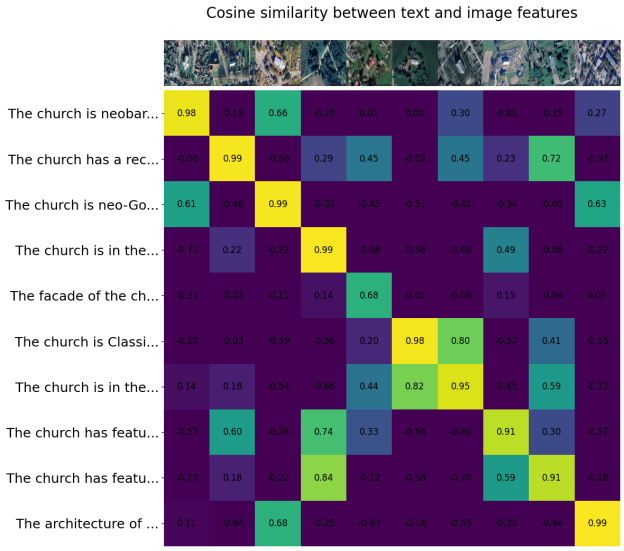

Figure 17. cosimilarity heatmap of token pooling with custom dataset

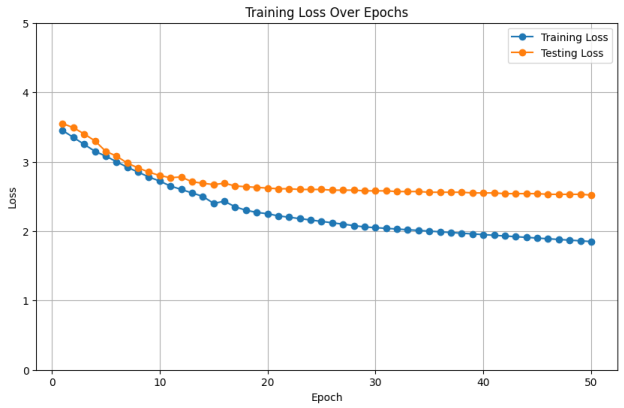

Figure 18. RSICD Loss curve with token pooling

14) shows slower training loss reduction, earlier plateauing of testing loss around epoch 10, and higher fluctuations, suggesting insufficient training duration or potential underfitting, perhaps an increase in epoch number can achieve better results. So a much larger dataset might be needed for a better model performance or perhaps there are problems with the custom dataset that might need reevaluation or generally trying to increase the dataset size. The average cosine similarity of the RSICD with our model modification is 65 percent.

Step 2. Amount Testing

When experimenting with the custom dataset of Lithuanian churches and training the model using token pooling, the results were underwhelming (Table 6). This method condensed input text into shorter prompts, which appeared to sacrifice context and reduce the model's ability to capture nuanced architectural descriptions. For example, the query "The church is rectangular in plan, with two towers" achieved an accuracy of 73%, reflecting the prevalence of basic rectangular plans in Lithuanian church architecture. However, even for such straightforward prompts, the model failed to retrieve accurate results for nearly a third of the images.

When tested with the more complex query, "The church has a classical, neoclassical, or classicist design," the accuracy dropped drastically to 15%. This significant decline underscores the model's inability to effectively generalize or recognize nuanced architectural styles with limited

input context. Similarly, for "The church is an example of Baroque architecture," the token pooling model struggled, achieving only an 18% accuracy rate, despite the dataset containing over 150 Baroque churches. These results highlight the model's difficulty in correlating stylistic details solely from roof-level satellite images.

The consistent underperformance suggests that the token pooling approach may be unsuitable for tasks requiring detailed comprehension of architectural styles. While it maintains reasonable accuracy for simpler queries, such as recognizing rectangular plans, it struggles with more intricate prompts. Thus, the limited input context imposed by token pooling impairs the model's ability to generalize and make precise architectural predictions.

Query	**Total Image Count**	**Incorrect Images**	**Accuracy Percentage**
The church is rectangular in plan, with two towers.	100	27	73
The church has a classical, neoclassical, or classicist design.	100	85	15
The church is an example of Baroque architecture.	100	82	18

Table 6. Table of queries experiments on Model trained with token pooling

The second fine-tuned model, trained with token pooling on the ChatGPT dataset, also showed limited success (Table 7). For simple queries, such as "rectangular plans," it achieved a maximum accuracy of 48%, which is comparable to the custom dataset's performance for similar tasks. However, it outperformed the custom dataset model in recognizing more intricate architectural styles.

For instance, the accuracy for identifying classical, neoclassical, or classicist designs was 34%, significantly higher than the 15% accuracy achieved with the custom dataset. Similarly, for Baroque architecture, the model achieved an accuracy of 31%, outperforming the custom dataset's token pooling model, which only managed 18%. These results suggest that the ChatGPT dataset may offer some advantages for broader recognition tasks when combined with token pooling.

Query	**Total Image Count**	**Incorrect Images**	**Accuracy Percentage**
The church is rectangular in plan, with two towers.	100	52	48
The church has a classical, neoclassical, or classicist design.	100	66	34
The church is an example of Baroque architecture.	100	69	31

Table 7. Table of queries experiments on Model trained with token pooling and Chat GPT outputs

The third fine-tuned model, trained with token pooling on the opensource RSICD dataset. For this dataset since the text data is different, the architectural styles will not be noticed, so the text

prompts will be changed (Table 8.) as we can see the second and third prompts have been changed which hold comparatively decent results (55 and 45).

query	total image count	incorrect images	accuracy percentage
The church is rectangular in plan, with two towers.	100	30	70
Church is next to a forest	100	45	55
Church is surrounded by buildings	100	55	45

Table 8. table of queries experiments on Model trained with token pooling and RSICD dataset

Step 3: CLIP explainability testing

The attention heatmap in Figure 19 illustrates the model trained with **token pooling**, tested with the prompt: *"A church with a rectangular plan, the church has two towers."*, you can see how it starts to effect the model's attention mechanism. The model takes every building into consideration where its attention should be focused on, considering every word in a sentence is considered as a seperete context token normally, we can expect the model to associate it easier, but in case of token pooling it averaging groups of tokens, effectively down sampling by combining the information within neighboring tokens. Thus the result may not be as focused and more dispersed than comparing the heat map of the model provided in (Figure 17)

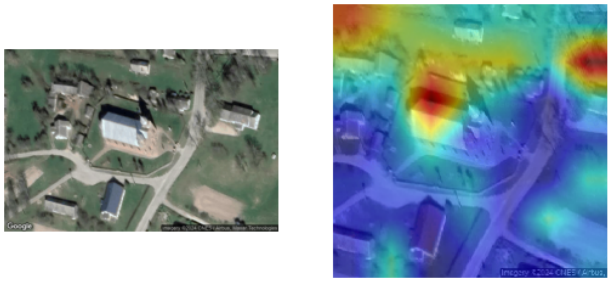

Figure 19. attention heatmap of the Model trained with token pooling. Prompt: A church with a rectangular plan, the church has two towers

Providing a different picture and prompts to the model trained with token pooling, the attention is completely off (Figure **??**). The model does not maintain any form of coherency when looking at a heat map. Mainly focusing on the forest near the church completely ignoring the given prompt, not only that any prompt in combination with this image provides similar or the same results.

If we provide the same picture and the same prompt to the model trained with token pooling (just like in Figure 19), however, trained on the ChatGPT dataset. We will see that the attention is very inaccurate, the attention heatmap is heavily dispersed and the focus is not on the church any longer, but on surrounding areas. (Figure 21). Such a low accuracy means that an auto-generated prompt description is not usable. Perhaps providing a more specific prompt to ChatGPT could be more beneficial, for example as a requirement to include the architectural style, emphasizing to base the description more on wikipedia sources and keeping more of a maintained structure in the prompts. Of course, the culprit for such a result will also be the small dataset needed to work with.

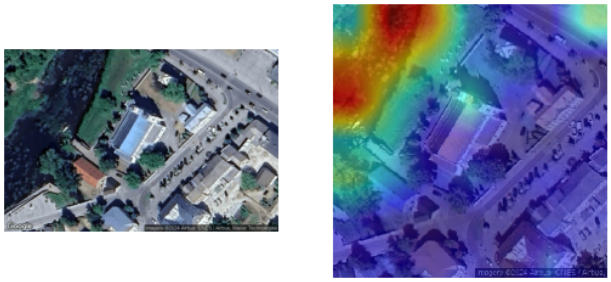

Figure 20. attention heatmap of the Model trained with token pooling. Prompt: The church has features of Baroque and historicism, with a rectangular plan (40 × 30 m), and five extensions, featuring two towers. The courtyard fence is made of plastered masonry.

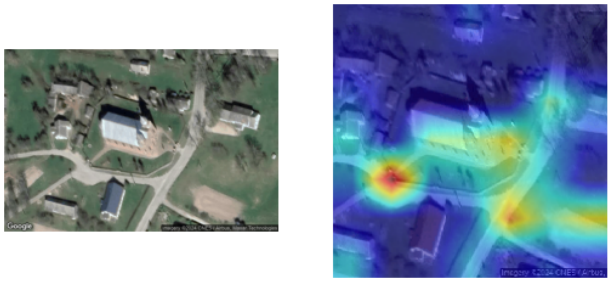

Figure 21. attention heatmap of the Model trained with token pooling on a ChatGPT description dataset. Prompt: A church with a rectangular plan, the church has two towers

query	**attention**	**image**	**model**
A church with a rectangular plan, the church has two towers	dispersed (hevy confusion with other buildings)	Šačių Šv. Jono church	token pooling
The church has features of Baroque and historicism, with a rectangular plan (40 × 30 m), and five extensions, featuring two towers. The courtyard fence is made of plastered masonry.	dispersed (hevy confusion with other objects)	Pasvalio Šv. Jono Krikštytojo church	token pooling
A church with a rectangular plan, the church has two towers	very dispersed (attention is not on the main building at this point)	Šačių Šv. Jono church	token pooling with ChatGPT description dataset

Table 9. attention heatmap experiments with finetuned models

The table (Table 9.) shows how experimentation with a custom datasets and different methods of trained models and how different attention is split between different models. So far the model with token pooling performs worse.

3.2.3 CLIP-longcontext

When trying to increase the description limit to the context length of 154 and using that token length to train the data on the same custom dataset we will get a much different result in terms of similarity. If we take a look in the top 4 resulting pictures (Figure 22) by giving the prompt "The church is classical, rectangular in plan, with two towers." we will see that the similarity rating goes up to 62.51 percent. Although, the images in figure 20 are very different in comparison to figure 9, so it is very curious how much an increased context length might effect the model.

Please note that these are the results before applying proper fine tuning.

Figure 22. Top 4 images of the caption "The church is classical, rectangular in plan, with two towers." Model trained with increased context length

Figure 23. Top 4 images of the much longer caption "The new brick church stands without the planned 33-meter high bell tower. A small, metal-framed bell tower has been erected in the churchyard. The church's tower is topped with a stainless steel cross on a rectangular base pedestal. The volume of the church building is modest, featuring a gable roof and a rectangular plan that harmonizes with the overall urban landscape of the settlement...". Model trained with increased context length

Now the question is why is the similarity rate so different. The hypothesis I assume is because of the shorter context provided in the prompt, or perhaps the reason for that is I was using the Hugging face architecture of (openai/vit-B32) instead of the git hub distribution of the CLIP model. So what will happen if we give a much longer text prompt that would fit closer to the context length limit of 154? For example: "The new brick church stands without the planned 33-meter high bell tower. A small, metal-framed bell tower has been erected in the churchyard. The church tower is topped with a stainless steel cross on a rectangular base pedestal. The volume of the church building is modest, featuring a gable roof and a rectangular plan that harmonizes with the overall urban landscape of the settlement. The building's architectural expression is characterized by elements typical of East Prussian neo-Gothic style, which dominate its exterior. The facade surfaces have minimal finishing, highlighting the central composition of brick texture and smooth

plaster, accentuated by a lightly covered main entrance". (Figure 23)

This prompt oddly enough was lower in terms of similarity ranking, dropping the similarity rate to 38.87 percent (Figure 23). Looking through the loss rate it is also very likely that that the model is suffering from overfitting. The previous in CLIP-longcontext segment results were achieved with base recommended parameters of vit/B32 finetuned parameters, however, the parameters might be somewhat better if we apply parameter optimizations.

Results for Hyperparameter finetuning (Table 10):

Trial Name	Status	LR	Batch Size	Weight Decay	Warmup Epochs	Iter	Total Time (s)	Loss	Val Loss	Top1 Acc %	Top5 Acc %
train_clip_f5a11_00000	TERMINATED	4.31565e-05	32	0.000530323	3	20	1868.94	2.70316	2.02388	18.6667	93.3333
train_clip_f5a11_00001	TERMINATED	0.000200748	32	0.000593131	7	2	194.972	3.41444	2.69247	4	20
train_clip_f5a11_00002	TERMINATED	7.47659e-05	32	0.000170632	5	2	194.713	3.41063	2.67378	2.66667	13.3333
train_clip_f5a11_00003	TERMINATED	0.000209669	32	0.000474681	4	2	196.064	3.41215	2.68796	2.66667	13.3333
train_clip_f5a11_00004	TERMINATED	0.000773126	32	0.000483691	7	4	378.569	3.43105	2.7086	1.33333	6.66667
train_clip_f5a11_00005	TERMINATED	0.000166874	32	0.000300566	5	2	196.27	3.41117	2.689	2.66667	13.3333
train_clip_f5a11_00006	TERMINATED	0.000123474	32	0.000283364	5	8	769.886	3.32544	2.58669	2.66667	13.3333
train_clip_f5a11_00007	TERMINATED	0.000110706	32	0.00019456	5	2	194.803	3.40688	2.67608	1.33333	6.66667
train_clip_f5a11_00008	TERMINATED	4.2175e-05	32	0.000192756	3	2	193.557	3.40745	2.68701	1.33333	6.66667
train_clip_f5a11_00009	TERMINATED	0.000272083	32	0.000116384	6	2	197.199	3.41245	2.68871	2.66667	13.3333
train_clip_f5a11_00010	TERMINATED	3.79976e-05	32	0.000465199	3	2	195.426	3.40976	2.6832	1.33333	6.66667
train_clip_f5a11_00011	TERMINATED	0.000104791	32	0.000339853	6	16	1522.71	3.0072	2.29578	10.6667	53.3333
train_clip_f5a11_00012	TERMINATED	0.000632679	32	0.00049084	7	8	765.042	3.38907	2.61462	4	20
train_clip_f5a11_00013	TERMINATED	3.72866e-05	32	0.000765	3	20	1860.09	2.6925	1.925	20	100
...											

Table 10. Training Results for CLIP Model

This is the result of the best finetuned model which is train_clip_f5a11_00013. Through the entire training trials this has brought us the best performance after 20 epochs. with the validation loss value of 1.9 after 19 epochs (Figure 24).

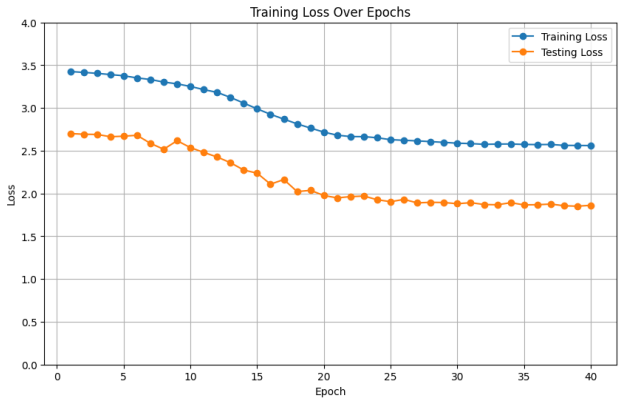

Figure 24. Finetuned clip model with custom dataset and increased context length

Training Loss:

The training loss decreases rapidly until epoch 20 and then plateaus around 2.6. This indicates that the model has learned most of what it can from the data, and further training does not reduce training loss significantly.

Testing Loss:

The testing loss decreases steadily until epoch 20, stabilizes around 1.9, and remains constant. This suggests the model's generalization capability has also saturated, and no further improvement occurs beyond epoch 20. Gap Between Training and Testing Loss:

There is a consistent gap between training and testing loss (approximately 0.6), which is normal and shows that the model generalizes reasonably well.

No Overfitting:

The testing loss does not increase, which means the model is not overfitting even at 40 epochs.

After the tests, it seems that the optimized model has an average cosine similarity of 80 percent. Which is much higher than the token pulled variant of 60 percent.

The clip model works, it tries to have cosimilarities that are of the same text image pair to be more similar to one another, and widen the gap between pairs that don't match, in other words it's called contrastive learning (Figure 25).

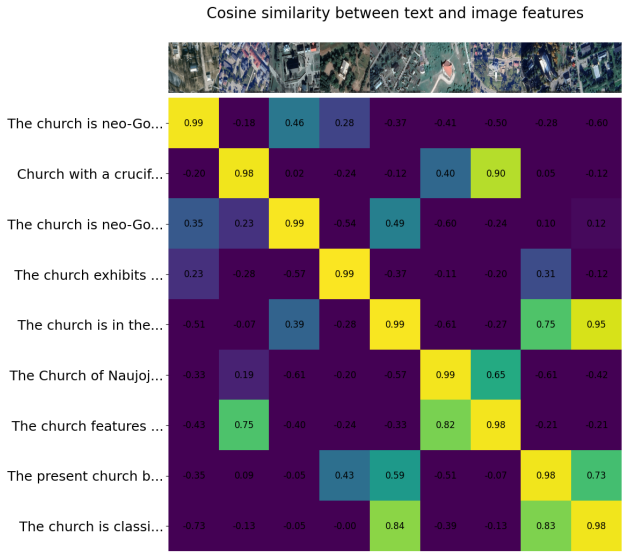

Figure 25. cosimilarity heatmap of long context

The heatmap (Figure 25) shows a strong alignment between text and image features for matching pairs (high diagonal values). The model successfully separates most non-matching pairs, but some non-diagonal cells show moderate similarity, indicating room for further improvement, possibly through hard negative mining.

The values range from approximately -1.0 to 1.0:

1.0 (bright yellow): Indicates very high similarity. Close to 0 (dark purple): Indicates little to no similarity. Negative values: Represent dissimilarity or oppositely aligned embeddings. Diagonal values (from top-left to bottom-right) are mostly bright yellow (0.99 or close to it):

This indicates that each text description has the highest cosine similarity with its corresponding image. This is expected in a well-aligned model like CLIP. Off-diagonal cells (non-matching pairs) have lower values, ranging from negative values to small positives:

This indicates the embeddings of non-corresponding text-image pairs are not as similar, which is good for alignment.

These are the top 4 results after finetuning (Figure 26 and Figure 27):

The two figures illustrate the top 4 image matches retrieved by a model trained with increased context length after hyperparameter fine-tuning. The caption is shorter and describes the church as "classical, rectangular in plan, with two towers." The similarity percentages are relatively high, with the top result achieving 90.55% similarity. The model performs well at retrieving images that match the concise description, showing effective alignment with the shorter textual context. The longer description that includes intricate details about the church, such as the "planned 33-meter high bell tower" and "stainless steel cross on a rectangular base pedestal." The model has

Figure 26. Top 4 images of the caption "The church is classical, rectangular in plan, with two towers." Model trained with increased context length after hyperparameter finetuning

Figure 27. Top 4 images of the much longer caption "The new brick church stands without the planned 33-meter high bell tower. A small, metal-framed bell tower has been erected in the churchyard. The church's tower is topped with a stainless steel cross on a rectangular base pedestal....". Model trained with increased context length after hyperparameter fine-tuning

no way of recognizing such details. The model appears to handle the extended context well and find the appropriate image from the dataset, but may struggle with over-specific or highly detailed descriptions. The results highlight that hyperparameter fine-tuning and increased context length improve the model's ability to align text and image data. However, shorter captions tend to produce slightly more accurate matches image-wise.

However, we can not change parameters such as hidden size, number attention heads, number of hidden layers, because that requires to train the model from scratch. And to train an entire new CLIP model we do not have enough data.

What we can also test is what will happen if we increase the Dataset size with image augmentation.

The augmentation must make the picture different, but not too different to confuse the model. So the method of augmentation chosen was rotation of up to 15 degrees, slight gausian blur of x and y values up to 10 percent, and slight color jitter values to affect brightness. (Figure 28. and Figure 29.)

Figure 28. unaugmented image

Figure 29. augmented image

Table 11. Hyperparameter Optimization Results and Best Configuration on data augmentation

Trial	Status	LR	Batch Size	Weight Decay	Dropout	Warmup Epochs	Loss	Val Loss	Top-1 Acc (%)	Top-5 Acc (%)
train_clip_7926c_00000	TERMINATED	2.11e-5	64	0.00016	0.0776	3	3.4484	2.7298	26.67	88.89
train_clip_7926c_00001	TERMINATED	3.24e-5	32	0.00021	0.1069	7	2.7834	2.7122	11.33	56.67
train_clip_7926c_00002	TERMINATED	2.49e-5	64	0.00050	0.1078	3	3.4348	2.6978	24.44	77.78
train_clip_7926c_00003	TERMINATED	2.24e-5	64	0.00022	0.0661	3	3.4855	2.7666	16.67	61.11
train_clip_7926c_00004	TERMINATED	2.84e-5	64	0.00012	0.1073	5	3.4799	2.8099	16.67	56.67
train_clip_7926c_00005	TERMINATED	2.06e-5	32	0.00067	0.0885	5	2.9666	2.8801	4.67	23.33
train_clip_7926c_00006	TERMINATED	2.24e-5	32	0.00050	0.0536	5	2.9511	2.9695	5.33	26.67
train_clip_7926c_00007	TERMINATED	3.50e-5	64	0.00014	0.0644	5	3.4270	2.6726	24.44	80.00
train_clip_7926c_00008	TERMINATED	2.58e-5	64	0.00032	0.1055	7	3.7193	2.9128	3.33	17.78
train_clip_7926c_00009	TERMINATED	3.31e-5	64	0.00038	0.0906	5	3.6493	2.9829	10.00	36.67

The best performance was attemp no 7 according to Validation loss for 64 Batch sizes. And attempt no 1 was better for 32 batch size (Table 12).

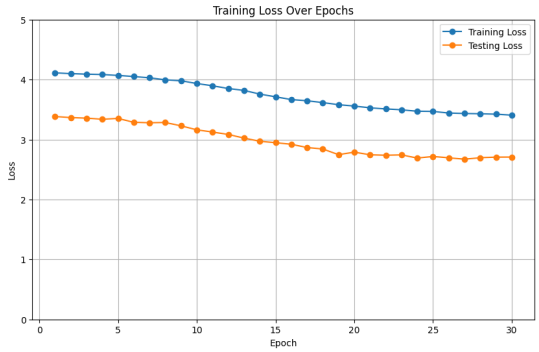

Figure 30. Loss curve on augmented dataset with a 64 batch param

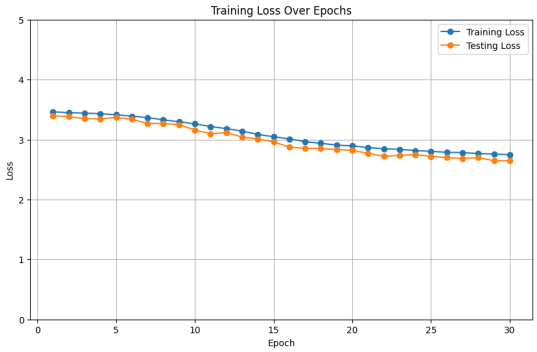

Figure 31. Loss curve on augmented dataset with a 32 batch param

The comparison of training and testing loss curves for batch sizes 32 and 64 on the augmented dataset reveals distinct differences in model performance. For a batch size of 64 (Figure 30), the training loss decreases gradually but stabilizes at a higher value, while the testing loss plateaus with a noticeable gap from the training loss, indicating potential underfitting. In contrast, for a batch size of 32 (Figure 31), both the training and testing losses decrease steadily and converge to lower values with a smaller gap between them, suggesting better optimization and generalization. The smaller batch size enables faster convergence due to more frequent gradient updates, resulting in improved overall performance. Thus, a batch size of 32 is more effective for training on the augmented dataset. The average cosine similarity is 75% for the 64 batch variant and 83% batch for the 32 batch variant.

This is the tuned clip model with increased context length of the RSICD dataset (Figure 32). This loss curve represents a training process over 50 epochs where both the training and testing losses decrease steadily, showing effective learning and generalization. The training loss starts at around 3.0 and decreases smoothly, plateauing after epoch 40 around a value near 1.8. The testing loss follows a similar trajectory but remains slightly higher than the training loss throughout, indicating a small but consistent generalization gap. The rapid descent between epochs 10 and 25 suggests efficient learning during this period, with stabilization occurring after epoch 30. It seems the Loss can decrease even further, It might be a good idea to increase the epoch length. The average cosine similarity of the RSICD dataset with our model changes is about 86 percent.

Step 2: Amount Testing

In contrast, the model trained with increased context length, allowing for longer and more descriptive prompts, exhibited improved performance across all tested queries (Table 12). For the

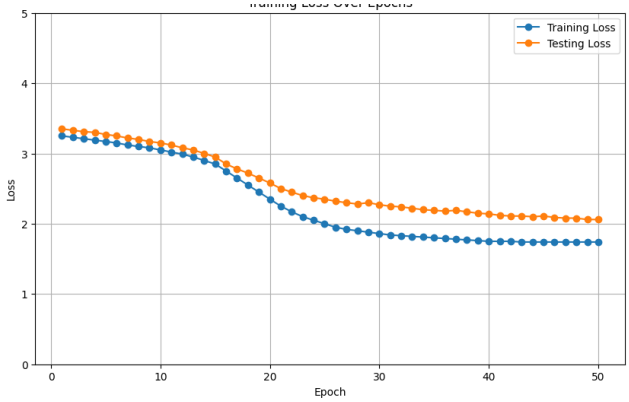

Figure 32. increased context length loss of RSICD

query "The church is rectangular in plan, with two towers," the accuracy was slightly lower (68%) than the token pooling model. This marginal drop suggests that while increased context length might not significantly impact simpler tasks, it does not hinder performance either.

However, for more intricate prompts, the benefits of increased context length become evident. The accuracy for "The church has a classical, neoclassical, or classicist design" rose to 40%. Similarly, for "The church is an example of Baroque architecture," the model achieved a 30% accuracy rate. These results indicate that longer input prompts enable the model to capture more nuanced details, leading to better architectural style recognition.

The improved performance, particularly for complex queries, suggests that providing more descriptive input allows the model to make better use of the available data. However, despite this improvement, the overall results remain suboptimal due to the limited size and diversity of the dataset. Thus, while increased context length is beneficial, further enhancements, such as expanding the dataset, are crucial to unlocking the full potential of this approach.

Query	**Total Image Count**	**Incorrect Images**	**Accuracy Percentage**
The church is rectangular in plan, with two towers.	100	32	68
The church has a classical, neoclassical, or classicist design.	100	60	40
The church is an example of Baroque architecture.	100	70	30

Table 12. Table of queries experiments on Model trained with increased context length

To analyze how the performance differs when using a ChatGPT-generated dataset, the same queries were tested with a fine-tuned model trained with increased context length (Table 13). The results indicate that the model's accuracy ranged from 29% to 51% across various queries. While these figures are lower than those obtained using the custom dataset for straightforward tasks like "rectangular plans," they are comparable for more intricate queries involving Baroque and classical architecture.

This lower performance suggests that the ChatGPT dataset, despite its potentially broader

scope, lacks the specificity needed for fine-grained architectural recognition. For example, the accuracy for recognizing Baroque architecture was 29%, slightly better than token pooling with the same dataset but significantly worse than models trained on the custom dataset with increased context length (30%). Similarly, the accuracy for classical designs stood at 36%, which, while better than some results from the custom dataset, still highlights the dataset's limitations.

Query	Total Image Count	Incorrect Images	Accuracy Percentage
The church is rectangular in plan, with two towers.	100	64	51
The church has a classical, neoclassical, or classicist design.	100	64	36
The church is an example of Baroque architecture.	100	71	29

Table 13. Table of queries experiments on Model trained with increased context length and Chat GPT outputs

The custom dataset demonstrates superior performance for simple architectural queries but struggles with more complex prompts due to its limited size and training data diversity. Increasing context length improves accuracy for complex prompts, particularly when distinguishing architectural styles. The ChatGPT dataset, while showing promise for general tasks, lacks the precision needed for fine-tuned architectural recognition. Expanding the custom dataset and refining the context length further could bridge the gap between accuracy and generalization, enhancing the model's utility for both simple and intricate queries.

The third fine-tuned model, trained with increased context length on the opensource RSICD dataset. For this dataset since the text data is different, the architectural styles will not be noticed, so the text prompts will be changed (Table 14.) as we can see the second and third prompts have been changed. And of course we can see a much better result while using the RSICD dataset.

query	total image count	incorrect images	accuracy percentage
The church is rectangular in plan, with two towers.	100	20	80
Church is next to a forest	100	44	56
Church is surrounded by buildings	100	52	48

Table 14. table of queries experiments on Model trained with token pooling and RSICD dataset

Step 3: CLIP explainability testing

The attention heatmap in (Figure 33) illustrates the model trained with **increased context length**, tested with the prompt: *"A church with a rectangular plan, the church has two towers."* The results show that the model primarily focuses on the main building, which is the church, while

also considering other rectangular buildings to a lesser extent. This focused attention aligns with expectations from earlier evaluations of the increased context length model, which demonstrated higher accuracy and coherence for similar prompts.

The precision in this model's attention highlights its ability to interpret and weigh specific architectural features effectively. This behavior is likely due to its ability to process and correlate long-context prompts better, preserving the details necessary for accurate image-query associations.

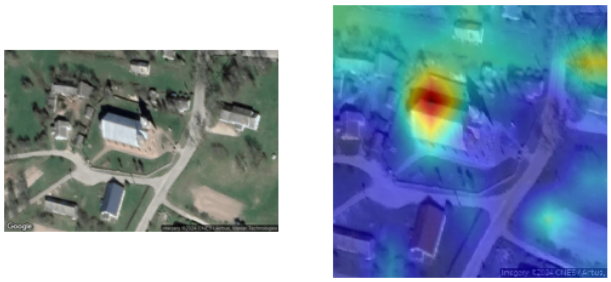

Figure 33. Attention heatmap of the model trained with increased context length. Prompt: A church with a rectangular plan, the church has two towers.

The attention heatmap (Figure 34) highlights two primary areas: the church structure and portions of the courtyard. The model focuses on regions that likely correspond to the described architectural features, including the towers and the rectangular plan of the church. This suggests that the model effectively associates the text's spatial and architectural descriptions with the image. Some parts of the image where focused on surrounding building attached to the masonry wall.

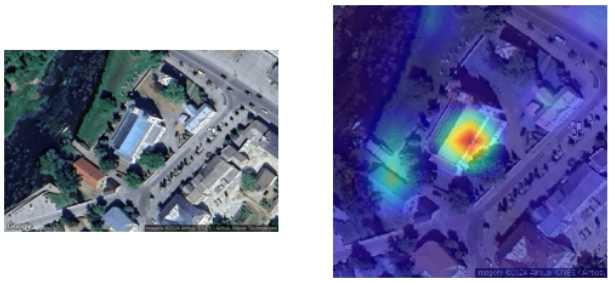

Figure 34. Attention heatmap of the model trained with increased context length. Prompt: The church has features of Baroque and historicism, with a rectangular plan (40 × 30 m), and five extensions, featuring two towers. The courtyard fence is made of plastered masonry.

The heatmap (Figure 35) shows a strong focus on a centralized area, which aligns with the location of the church. The model also emphasizes parts of the courtyard, potentially indicating its recognition of the walled courtyard described in the prompt. This suggests that the model's attention captures the unique structural elements described, such as the sacristy and the single tower. However not the entire cemetery was captured, indicating the model focuses more on the main building than surrounding objects.

The model primarily focuses its attention towards the main building (Table 15), which is the church. We can expect that to happen because of the amount of detail wikipedia descriptions provide on the main building itself.

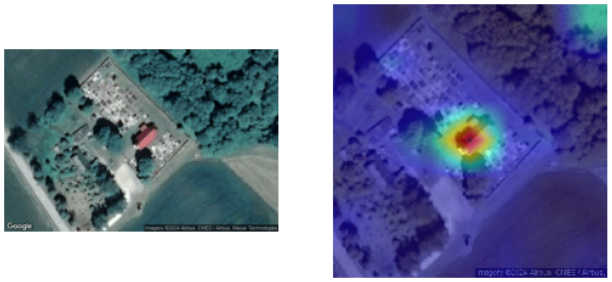

Figure 35. Attention heatmap of the model trained with increased context length. Prompt: The church is neo-Gothic, with a rectangular plan, featuring an annex for the sacristy, and has a single tower. The courtyard is walled. It contains a cemetery.

query	**attention**	**image**	**model**
A church with a rectangular plan, the church has two towers	Centered, slight focus on surroundings	Šačių Šv. Jono church	increased context with custom dataset
The church has features of Baroque and historicism, with a rectangular plan (40 × 30 m), and five extensions, featuring two towers. The courtyard fence is made of plastered masonry.	heavily centered on main building, some attention is noticed on surrounding buildings, walls	Pasvalio Šv. Jono Krikštytojo church	increased context with custom dataset
A church with a rectangular plan, the church has two towers	very centered	Žukančių Nukryžiuotojo Jėzaus church	increased context with custom dataset

Table 15. attention heatmap experiments with finetuned models

3.3 Comparison of Results

Model Configuration	**Accuracy form top 100 avg**	**Attention Focus**
Pretrained CLIP	79%	Good for simple features, dispersed for complex ones
Token Pooling	35%	Dispersed focus; struggles with details
Increased Context Length	46%	slightly more Precise focus than token pooling

Table 16. Comparison of Experimental Results

The experimental results highlight notable differences in performance across the tested configurations of the CLIP model (Table 16): the pre-trained open-source model, the token pooling fine-tuned model, and the model with increased context length. The pre-trained CLIP model achieved

an average top 100 accuracy of 79%, demonstrating reasonable alignment for simpler queries but struggled with more complex descriptions. Token pooling, while computationally efficient, resulted in a decreased average top 100 accuracy of 35% and exhibited limitations in attention coherence and detail recognition, as evidenced by dispersed attention heatmaps and a significant drop in accuracy for complex architectural queries. In contrast, the model trained with increased context length achieved average top 100 accuracy of 46% and showed better performance in identifying nuanced architectural styles such as Baroque and Neoclassical, however this might be the case of the model better memorizing captions rather than generalizing.

experiment	**batch size**	**epoch**	**training loss**	**testing loss**	**avg co sim**
CLIP-opensource dataset with pre-trained model	32	20	1.9	2.6	68
Clip-token pooling with custom dataset	32	20	2.8	1.9	61
Clip-token pooling with RSICD dataset	32	40	1.9	2.63	65
Clip-long context with custom dataset	32	40	2.69	1.9	80
Clip-long context with increased custom augmented train dataset with batch 64	64	30	3.4	2.7	75
Clip-long context with increased custom augmented train dataset with batch 32	32	30	2.78	2.7	83
Clip-long context with RSICD dataset	32	50	2.1	1.8	86

Table 17. table of resulting model configurations tested

The experimental results summarized in (Table 17) provide a comparative analysis of various configurations tested using the CLIP model across different datasets, training strategies, and hyperparameter settings. The baseline model, using the open-source pre-trained CLIP model with a batch size of 32 and 5 epochs, achieved an average cosine similarity of 68.5% with a training loss of 1.9 and a testing loss of 2.6, demonstrating a reasonable starting point but limitations in handling nuanced tasks. Token pooling on a custom dataset for 20 epochs resulted in a reduced cosine similarity of 61%, indicating challenges in maintaining semantic alignment, despite achieving a lower testing loss of 1.9. Extending token pooling to the RSICD dataset for 40 epochs yielded an improvement to 65% cosine similarity, suggesting better alignment with domain-specific data.

Notably, introducing a long-context approach on the custom dataset significantly improved performance, achieving 80% cosine similarity with a testing loss of 1.9 after 40 epochs. Further augmentation of the training dataset, combined with a larger batch size of 64, saw a slight decrease to 75%, highlighting the trade-offs between batch size and generalization. However, using a smaller batch size of 32 with the same augmented dataset improved performance to 83%, underlining the

importance of batch size in maintaining fine-grained learning. Finally, applying the long-context approach to the RSICD dataset over 50 epochs resulted in the highest cosine similarity of 86%, coupled with the lowest testing loss of 1.8, demonstrating the effectiveness of domain-specific fine-tuning and the long-context approach in improving model performance on detailed and descriptive tasks. These results emphasize the role of dataset augmentation, batch size, and context length in optimizing the CLIP model's ability to align image and text representations.

The CLIP explainability tests in the pre-trained model showed relatively focused attention on specific features mentioned in the text prompt, such as "aircraft" or "highways," albeit with limitations for complex queries like architectural descriptions. This indicates a good but basic alignment capability.

For models fine-tuned with token pooling, attention heat maps showed dispersed and less coherent focus. This method, which averages neighboring tokens, diluted the attention mechanism and reduced its ability to focus on critical features, particularly in complex architectural contexts. In contrast, the models fine-tuned with increased context length demonstrated somewhat improved attention. These models had the capacity to capture intricate details from longer text prompts, producing more accurate and focused heat maps. The increased context allowed for more detailed descriptions and better alignment with images, highlighting their potential for tasks requiring nuanced understanding. However, the specific details from Wikipedia descriptions makes the model focus heavily on the church itself rather then the surrounding environments.

Overall, while the token pooling approach struggled with maintaining attention granularity, the increased context length enabled more precise and coherent focus, making it better suited for explainability and detailed tasks. However, both methods underscore the importance of dataset size and quality in achieving optimal results.

4 Conclusion

Overall, the integration of Vision LLMs like CLIP in satellite imagery analysis has profound implications for geolinguistical search systems. By iteratively enhancing model architectures and training methodologies, we can unlock further potential in these tools for tackling intricate and domain-specific challenges in image understanding. The literature analysis and preparation of the methodology were crucial in laying the foundation for the Contrastive Language-Image Pre-training model training specifics. Furthermore, the experiments demonstrate the importance of tokenization step in clip and how much it can affect the training model and potential improvements that might be applicable to enhance the trained Clip models.

The experimental results underscore the impact of extended context length and dataset customization in achieving superior alignment and descriptive accuracy. Models fine-tuned with increased context length demonstrated notable improvements in capturing nuanced architectural details and producing coherent attention heatmaps. Conversely, token pooling strategies, while computationally efficient, exhibited limitations in granularity and detail recognition.

Key findings from the experiments underscore that:

- Models fine-tuned with increased context length achieved notable improvements, with cosine similarity reaching up to 86% on the RSICD dataset and 83% on augmented custom datasets. These models excelled in identifying nuanced architectural features but often over-focused on main objects, like churches, due to overly detailed text descriptions.

- Token pooling, while computationally efficient, exhibited a significant drop in performance, with an average cosine similarity of 61% and dispersed attention, limiting its ability to generalize and capture intricate details in architectural queries.

- The pre-trained CLIP model performed reasonably well on simpler tasks, achieving an accuracy of 79% for top 100 results, but struggled with complex queries requiring finer detail recognition.

Notably, the model trained on custom datasets, including Lithuanian church imagery, performed acceptably for complex prompts. However, challenges with dataset size and specificity were evident, particularly for highly detailed or stylistically nuanced queries. The experiments also emphasized the importance of balancing training parameters such as batch size, dataset augmentation, and context length for optimal performance.

It must be mentioned That the custom dataset may not be that effective in comparison to the open source dataset. The reasoning for such is not only the smaller dataset in comparison to RSICD, but also the provided detail from wikipedia descriptions. Hence the description is not only very long and verbose for a clip model, it provides too much detail for small features about the church exterior architecture. And we can see that with the long context clip model's experiment about explainability, the amount of detail provided by the text will make the model focus a lot more on the church itself rather than the surrounding details.

Manual analysis of incorrect images from semantic search revealed instances where the model's performance deviated from expectations. Unexpected images often included images that lacked the expected features described in the query.

CLIP explainability tests provided insights into how the model perceives and interprets images based on text prompts. Attention heatmaps demonstrated the model's focus on specific features

mentioned in the text. Token pooling proved to have adverse effects, not generalizing the image as effectively.

When fine-tuned on a custom dataset of Lithuanian churches, models trained with increased context length showed notable improvements in accuracy for complex prompts. These models effectively identified key architectural features like "rectangular plan" and "two towers." However, they still faced challenges with nuanced architectural styles such as Baroque or Neoclassical due to the limited dataset size, or the dificulty for the model to detect small details.

The use of token pooling as a training strategy proved less effective for complex queries, as it reduced the granularity of attention across key features. Heatmap analyses demonstrated that token pooling models had dispersed and less coherent attention, often focusing on irrelevant features or multiple objects. This finding underscores the trade-off between computational efficiency and model precision.

Nevertheless, the training of the model could benefit from more work and research. For one, a higher quality dataset with more data and descriptions that are more related with the surrounding environment seen on the image itself rather than focusing on the main building would help, allowing the model to generalize better instead of memorizing the descriptions to the related image. Further more, optimization strategies could be more effectively explored, and spending more time fine-tuning hyper parameters and applying better regularization could potentially help improve results even on a small dataset of Lithuanian churches. Other long text processing techniques like attention mechanism refinement (or attention pooling) might also help. Regardless, with even more improvements, the fine-tuned semantic search model may be used for more precise geographical hotspot finding (in our case churches) utilizing long, human like descriptions on the map of Lithuania.

Recommendation / future directions:

- Expanding and refining the dataset with more images and descriptions focusing on broader environmental details rather than isolated architectural features.
- Exploring advanced text processing techniques, such as attention pooling or enhanced positional embeddings, to further optimize performance with long textual inputs.
- Fine-tuning hyperparameters and applying robust regularization techniques to enhance model generalization, even on small datasets.

References

[1] B. L. Turner, D. Skole, S. Sanderson, G. Fischer, L. Fresco, and R. Leemans, “Land-use and land-cover change: Science/research plan”, 1995.

[2] Sherzod Hakimov and David Schlangen, "Images in Language Space: Exploring the Suitability of Large Language Models for Vision and Language Tasks"

[3] Gengchen Maia, Yao Xuanb, Wenyun Zuoc, Yutong Hed, Jiaming Songd, Stefano Ermond, Krzysztof Janowicze and Ni Lao, "Sphere2Vec: A General-Purpose Location Representation Learning over a Spherical Surface for Large-Scale Geospatial Predictions"

[4] John Jacob, Shantanu Nair, "LLMs, a brief history and their use cases"

[5] Haotian Liu, Chunyuan Li, Yuheng Li, Yong Jae Lee, University of Wisconsin–Madison, Microsoft Research, Redmond, "Improved Baselines with Visual Instruction Tuning"

[6] Arto, Dev Vidhani, Goutham, Mayank Bhaskar Sujit Pal, "Fine tuning CLIP with Remote Sensing (Satellite) images and captions"

[7] Rohin Manvi, Samar Khanna, Gengchen Mai, Marshall Burke, David Lobell, Stefano Ermon, "GEOLLM: EXTRACTING GEOSPATIAL KNOWLEDGE FROM LARGE LANGUAGE MODELS"

[8] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever, "Learning Transferable Visual Models From Natural Language Supervision", 2021

[9] Zhenghang Yuan, Zhitong Xiong, Lichao Mou, and Xiao Xiang Zhu "ChatEarthNet: A Global-Scale Image-Text Dataset Empowering Vision-Language Geo-Foundation Models" 2024

[10] https://newsroom.ibm.com/2023-08-03-IBM-and-NASA-Open-Source-Largest-Geospatial-AI-Foundation-Model-on-Hugging-Face

[11] "https://cratedb.com/blog/llm-vector-database-what-is-a-vector-databases-for-llm"

[12] "https://medium.com/@naveenjothi040/semantic-search-with-llms-3661fd2a9331"

[13] "https://medium.com/@tenyks_blogger/how-to-build-an-image-to-image-search-tool-using-clip-pinecone-b7b70c44faac"

[14] Richard Souvenir, Stephen MacNeil, Albatool Wazzan, "Comparing Traditional and LLM-based Search for Image Geolocation", 2024

[15] Beichen Zhang, Pan Zhang, Xiaoyi Dong, Yuhang Zang, Jiaqi Wang "Long-CLIP: Unlocking the Long-Text Capability of CLIP", 2024 https://arxiv.org/pdf/2403.15378

[16] A. Gupta and M. Gupta, “Transfer learning for small and different datasets: Fine-tuning a pre-trained model affects performance,” Journal of Emerging Investigators, 2020.

[17] M. Cherti and J. Jitsev, “Effect of pre-training scale on intra- and inter-domain full and few-shot transfer learning for natural and medical x-ray chest images,’

[18] L. P. Osco, Q. Wu, E. L. de Lemos, W. N. Gon¸calves, A. P. M. Ramos, J. Li, and J. M. Junior, “The segment anything model (sam) for remote sensing applications: From zero to one shot,” International Journal of Applied Earth Observation and Geoinformation, vol. 124, p. 103540, 2023.

[19] Aakash Sikarwar, Aditya Pawar, Akhil Sethiya, Akshat Surana, Shivshankar Rajput, "CLIP Model Image Search: A Deep Learning Approach for Finding Relevant Images", 2022.

[20] Piotr Nawrot, Jan Chorowski, Adrian Łancucki, Edoardo M. Ponti, University of Edinburgh, "Efficient Transformers with Dynamic Token Pooling", 2023.

[21] UNESCO. "Atlas of the World’s Languages in Danger", 2010.

[22] Amery, R. "The Role of Geolinguistics in the Preservation of Endangered Languages". In Handbook of Endangered Languages. Routledge. 2019.

[23] Genčių genealogija "Digital map of churches" 2017
https://www.arcgis.com/home/item.html?id=feb35d29fa9f4582a71d728c77ef286e

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger and lya Sutskever, "Learning Transferable Visual Models From Natural Language Supervision", 2021
https://arxiv.org/pdf/2103.00020v1