

Simen Myhre Waitz

Automated Image Captions for Norwegian Real Estate Advertisements

Master's thesis in Cybernetics and Robotics

Supervisor: Ole Morten Aamo, NTNU

Co-supervisor: Ulf Jakob Flø Aarsnes, Solgt.no

June 2024

Simen Myhre Waitz

Automated Image Captions for Norwegian Real Estate Advertisements

Master's thesis in Cybernetics and Robotics
Supervisor: Ole Morten Aamo, NTNU
Co-supervisor: Ulf Jakob Flø Aarsnes, Solgt.no
June 2024

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics



Abstract

ABSTRACT

This project aims to develop an automated system for generating accurate captions in a natural Norwegian language for images in real estate listings. The research employs transformers, specifically utilizing the Bootstrapping Language-Image Pre-Training (BLIP) model, to achieve this objective. The criteria for the product is that it need to be open-sourced, as well as being time and energy efficient compared to manually writing or generating with other state-of-the-art image captioning models.

The dataset comprised images and corresponding captions from real estate advertisements written by realtors. The research involves extensive data preprocessing. This includes image classification, language cleaning, and caption classification using vision and sentence transformers as key components, and then balancing the data with K-means clustering. BLIP of different model sizes is then fine-tuned on preprocessed datasets consisting of kitchen images as a proof of concept.

Moreover, the models that are trained on different datasets and hyperparameters are evaluated using automatic and human evaluation metrics. A key component is to investigate and develop a suited evaluation framework for this open-ended generative task. The solution focuses on automatically calculating lexical diversity and textual similarity metrics on a test set. Additionally, a survey and user test were conducted to gather qualitative feedback on the generated captions compared to the ones written by realtors and GPT-4.

The research proves that with a suited dataset and optimal hyperparameters, BLIP can be fine-tuned to generate captions with similar quality as real estate agents. Concluding that this project provides an energy- and time-efficient solution to reduce the workload demanded in writing real estate advertisements.

SAMMENDRAG

Målet med prosjektet er å utvikle et automatisert system for å generere nøyaktig bildetekst med naturlig norsk språk for bilder i boligannonser. Forskningen benytter transformere, med fokus på Bootstrapping Language-Image Pre-Training (BLIP)-modellen, for å oppnå dette målet. Kriteriene for produktet er at modellen som tas i bruk har offentlig kildekode, samt være tids- og energieffektiv sammenlignet med andre KI-modeller eller å manuelt skrive bildebeskrivelsene.

Datasettet består av bilder og tilhørende bildetekster fra boligannonser skrevet av meglere. Forskningen innebærer omfattende forbehandling av datasettet, inkludert bildeklassifisering, filtrering av språk og tekstklassifisering ved bruk av bilde- og tekst transformatorer, og balansering av data ved hjelp av K-means clustering. Deretter finjusteres forskjellige modellstørrelser av BLIP på forhåndsbehandlede datasett bestående av kjøkkenbilder som et bevis på at prosjektets oppgave er gjennomførbart.

Videre blir modellene som er trent på forskjellige datasett og hyperparametre evaluert ved hjelp av automatisk utregnede og menneskelige evalueringsmetriker. En sentral komponent er å undersøke og utvikle en egnet evalueringsramme for oppgaven. Løsningen fokuserer på å undersøke metoder for å automatisk beregne kvalitet på ordforråd og likhet med megler-skrevet beskrivelser. I tillegg ble en undersøkelse og brukertest gjennomført for å samle kvalitativ tilbakemelding på de genererte bildetekstene sammenlignet med de som er skrevet av meglere og GPT-4.

Rapporten viser at BLIP kan finjusteres til å generere bildetekster på norsk med tilsvarende kvalitet som de som er skrevet av eiendomsmeglere og referansemodellen GPT-4. Konklusjonen er at dette prosjektet gir en energi- og tidsbesparende løsning for å redusere arbeidsmengden som kreves for å skrive boligannonser.

Preface

This is a master's thesis in Cybernetics and Robotics at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU). It is written during the months of February to June 2024. The project idea as well as results presented in the report, are made by the student, *Simen Myhre Waitz*.

A special thanks to Ole Morten Aamo at NTNU and Ulf Jakob Flø Aarsnes at Solgt.no for the collaboration and supervision. Also, thanks to all the good people who publish open-source machine learning algorithms online.

Contents

Abstract	i
Preface	iii
Contents	vi
List of Figures	vi
List of Tables	ix
Abbreviations	x
1 Introduction	1
1.1 Project description	1
1.1.1 Collaboration with Solgt.no	2
1.2 Motivation	2
1.2.1 Energy Efficiency using Smaller Models	2
1.2.2 Contribution to Norwegian Generative AI Research	2
1.2.3 Automation of Manual Tasks	3
1.3 Prerequisites	3
1.3.1 Hardware	3
1.3.2 Software	4
1.4 Report Structure	4
2 Preliminaries	5
2.1 A Simple Introduction to Neural Networks	5
2.2 Image Captioning	6
2.2.1 Foundation Models	6
2.3 Transformers	7
2.3.1 Vision Transformers	8
2.4 Convolutional Neural Network	9
2.4.1 CNN Architecture	9
2.5 State-of-the-art NLG	10
2.5.1 Hugging Face	10
2.5.2 GPT-4 by OpenAI	11
2.5.3 NoraLLM	11
2.5.4 Stable Diffusion 3 by Stability AI	11

CONTENTS

2.5.5	BLIP by Salesforce	11
2.6	Evaluation Metrics for Text Generation	13
2.6.1	Automatic Evaluation	13
2.6.2	Human Evaluation	16
2.7	ML for Norwegian Real Estate Listings	17
2.7.1	Finn.no	17
2.7.2	Room Classification by Hjem.no	18
2.7.3	Automatic Generated Prospects by Prosper AI	18
3	Data	19
3.1	Dataset	19
3.1.1	Data Format	19
3.2	Preprocess	21
3.2.1	Classify Images	22
3.2.2	Caption Cleaning	23
3.2.3	Classify Captions	23
3.2.4	Balance The Data with K-Means Clustering	23
3.3	Final Datasets	24
4	Method	25
4.1	Hyperparameters	25
4.1.1	Learning Rate	25
4.1.2	Epochs	26
4.1.3	Batch Size	26
4.2	Regularization	27
4.2.1	Train and Validation Split	27
4.2.2	Early Stopping	27
4.3	BLIP	27
4.3.1	BLIP License	28
4.3.2	Parameter Tuning Log for BLIP	28
4.4	Implementation of Automatic Evaluation Metrics	29
4.4.1	Average Sentence Length	30
4.4.2	TTR	30
4.4.3	Yule’s K	30
4.4.4	BLEU	30
4.4.5	STS	30
4.4.6	Overview of Metrics	30
4.5	Postprocess	31
4.6	Survey for Testing The Model	31
4.6.1	Question Segment One	31
4.6.2	Question Segment Two	32
4.6.3	Question Segment Three	33
4.6.4	Data Selection	34
4.7	Website for User Test	34
5	Results	36
5.1	Training BLIP	36
5.1.1	Training and Validation Loss	36
5.2	BLIP Trained on Non-preprocessed Data	38

CONTENTS

5.2.1	Multiple Room Types	38
5.2.2	Kitchen	39
5.3	Automatic Evaluation Results	40
5.3.1	Mixed Class Dataset	40
5.3.2	Descriptive and Creative Datasets	42
5.4	Developer’s Analysis	42
5.4.1	Mixed Class Dataset	42
5.4.2	Descriptive vs. Creative	44
5.5	Survey Results	45
5.5.1	Question Segment One	45
5.5.2	Question Segment Two	47
5.5.3	Question Segment Three	47
5.6	User Test Results	49
5.7	GPT-4	50
5.7.1	Inference Speed Comparison	50
5.7.2	Caption Comparison GPT-4 and BLIP	51
6	Discussion	52
6.1	Dataset	52
6.1.1	Comment on Preprocessing	52
6.1.2	Descriptive vs. Creative	53
6.2	Hyperparameters Selection	53
6.3	Automatic Evaluation	54
6.4	Human Evaluation	54
6.4.1	The Developer’s Evaluation	54
6.4.2	Survey Results Discussion	55
6.4.3	User Test	55
6.5	Comment on Best Performing Models	56
6.6	Limitations	56
6.6.1	GPU Limitations	56
6.6.2	User Test Limitations	56
6.7	Future Work	57
6.7.1	Dataset	57
6.7.2	Embedding	57
6.7.3	Extract Structured Data	57
7	Conclusion	58
	References	59
A	Examples of Generated Captions	65
B	Demonstration Videos of Website	71
C	Code Snippets for Automatic Evaluation	72

List of Figures

2.2.1	Foundation model illustration	7
2.3.1	ViT architecture	9
2.4.1	Simple illustration of a CNN [23]	10
2.5.1	Architecture of BLIP [30]	12
2.5.2	Cross-entropy loss	13
3.1.1	Finn.no image gallery example	19
3.1.2	Finn.no example one	20
3.1.3	Finn.no example two	20
3.2.1	Flowchart of Preprocessing Stages	22
4.2.1	Early Stopping	28
4.6.1	Survey segment one examples	32
4.6.2	Survey segment two example	33
4.6.3	Survey segment three example	34
4.7.1	Screenshot Captioner website	35
5.1.1	Loss with different LR	37
5.1.2	Loss with different Epochs	37
5.1.3	Loss with different L2 Regularization	38
5.2.1	Bedroom and Living Room captions	39
5.2.2	Bathroom and Kitchen captions	39
5.4.1	Examples with BLIP Large models	44
5.4.2	Example of Descriptive and Creative captions	45
5.5.1	Segment one: Preferred caption type	45
5.5.2	Segment one: "Identifying AI-generated caption" score	46
5.5.3	Segment two: Preferred caption type	47
5.5.4	Segment three: Caption-quality scores	48
5.5.5	Segment three: Average caption-quality scores	48
5.5.6	Segment three: Box plot caption-quality scores	49
5.7.1	Screenshot of usage cap error-message in GPT-4	50
5.7.2	GPT-4 and BLIP: Captions example one	51
5.7.3	GPT-4 and BLIP: Captions example two	51
A.0.1	Model ID 14, example one	65
A.0.2	Model ID 15, example one	65
A.0.3	Model ID 14, example two	66

LIST OF FIGURES

A.0.4	Model ID 15, example two	66
A.0.5	Model ID 22, example one	67
A.0.6	Model ID 22, example two	67
A.0.7	Model ID 20, example one	68
A.0.8	Model ID 22, example three	68
A.0.9	Model ID 20, example two	69
A.0.10	Model ID 22, example four	69
A.0.11	Model ID 20, example three	70
A.0.12	Model ID 22, example five	70

List of Tables

3.2.1 Image Categories	22
4.3.1 BLIP Training Log	29
5.1.1 Training Times BLIP	36
5.3.1 Automatic Scores: BLIP Base 90-10% split	40
5.3.2 Automatic Scores: BLIP Base 70-30% split	41
5.3.3 Automatic Scores: BLIP Large	41
5.3.4 Automatic Scores: Descriptive and Creative	42
5.4.1 Developer’s evaluation: BLIP Base 90-10% split	43
5.4.2 Developer’s evaluation: BLIP Base 70-30% split	43
5.4.3 Developer’s evaluation: BLIP Large	43
5.5.1 Identifying AI survey scores	46
5.6.1 User Test’s evaluation	50

Abbreviations

List of all abbreviations in alphabetic order:

- **AI** Artificial Intelligence
- **AVM** Automated Valuation Model
- **BERT** Bidirectional Encoder Representations from Transformers
- **BLEU** BiLingual Evaluation Understudy
- **BLIP** Bootstrapping Language-Image Pre-Training
- **BS** Batch Size
- **CLIP** Contrastive Language-Image Pre-Training
- **CNN** Convolutional Neural Network
- **CTR** Click-through Rate
- **DNN** Deep Neural Network
- **JPG** Joint Photographic Experts Group
- **LR** Learning Rate
- **GPT** Generative Pre-trained Transformer
- **GPU** Graphics Processing Unit
- **MED** Multimodal mixture of Encoder-Decoder
- **ML** Machine Learning
- **NHO** Confederation of Norwegian Enterprise
- **NLG** Natural Language Generation
- **NLP** Natural Language Processing
- **NN** Neural Network
- **NTNU** Norwegian University of Science and Technology

LIST OF ABBREVIATIONS

- **RAM** Random-access Memory
- **RGB** Red, Green and Blue
- **RNN** Recurrent Neural Network
- **STS** Semantic Textual Similarity
- **ViT** Vision Transformer
- **VLP** Vision-Language Pre-Training
- **VPN** Virtual Private Network

Introduction

This chapter is an introduction to the project’s objective, given the motivation and collaboration behind the report.

1.1 Project description

The official description of this project is as follows.

Train a neural network to automatically generate Norwegian captions for images in real estate listings.

Real estate advertisements in Norway contain images and captions explaining the real estate object. This is the foundation of a large dataset that can be utilized for fine-tuning neural networks to automatically generate Norwegian captions based on input images. The main objectives of this project are summarized in four points:

- Finding a small and energy-efficient deep neural network that can be trained to write automated image captions in Norwegian.
- Develop a process for constructing a dataset for optimal training.
- Tune the model and search for optimal hyperparameters.
- Find suited metrics and methods for evaluating and testing this specific Natural Language Generation (NLG) task.

The focus of this master’s thesis is specifically on images of kitchens rather than all types of images. This approach serves as a proof of concept and allows for more time to be dedicated to all stages of developing an automatic image captioning model. With this selected scope, the project can develop a method to preprocess the data to achieve clean, diverse, and balanced datasets, train an image captioning model on these datasets, and finally, conduct research on the evaluation framework and assess the results.

1.1.1 Collaboration with Solgt.no

This thesis is supervised and in collaboration with a Norwegian PropTech¹ company, Solgt.no [1]. Ulf Jakob Flø Aarsnes is a co-supervisor on this thesis and is the head of research and development at Solgt.no.

Solgt.no has developed an Automated Valuation Model (AVM) that automatically estimates the value of real estate. They have launched a website that provides users with access to a Decision Support System (DSS) for pricing dwellings and a tool for housing market analysis.

1.2 Motivation

The real estate industry is Norway’s largest asset class [2]. There are 3.1 million Norwegians annually browsing real estate listings on Finn.no. Each listing is currently manually produced, and the captions are written by real estate agents 97% of the time [3]. This presents a substantial opportunity to develop one of the first open-sourced Norwegian image captioning models with purpose of reducing the workload for real estate agents. There are three primary motivations for this development. It is to provide an automatic option for a repetitive task, as well as being an environmentally friendly product that also contributes to research within the field of natural language generative tasks in Norwegian.

1.2.1 Energy Efficiency using Smaller Models

While large foundation models like OpenAI’s GPT-4 are the benchmarks in image-to-text tasks, their use is costly and unsustainable for specific, repetitive tasks. These models require immense computational resources, leading to high energy consumption. For instance, GPT-4’s daily power usage is said to be comparable to that of 180,000 U.S. households [4].

Fine-tuning smaller models offers a more sustainable and cost-effective solution. These models are less resource-intensive, reducing the carbon footprint associated with GPU computations. Moreover, smaller models, when fine-tuned with specific datasets, can yield higher quality results by leveraging more relevant data. This approach aligns with the UN Sustainable Development Goal 9, which aims to build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation [5]. By adopting smaller, task-specific models, we can address sustainability concerns while maintaining high performance standards.

1.2.2 Contribution to Norwegian Generative AI Research

The reliance on large, international artificial intelligence (AI) models poses a significant issue for Norwegian AI development. When these models are used, valuable data and user test feedback are often sent to major American tech companies, which hinders local innovation and control. Developing Norwegian language generative models ensures that these critical resources remain within the country.

The Norwegian government recognizes the importance of AI research and has committed approximately one billion NOK to this field. The AI initiative has three main tracks,

¹Property Technology

whereas one is research on how digital technologies can be used for innovation in business and the public sector, and how artificial intelligence can be utilized in research across various disciplines [6]. This master’s thesis contributes to this national effort by building and disseminating generative AI models tailored for the Norwegian language and context. The goal is not to compete with global giants but to provide high-quality, localized models that serve the specific needs of Norwegian businesses.

1.2.3 Automation of Manual Tasks

Generative AI offers substantial economic benefits by increasing productivity, as noted by the Confederation of Norwegian Enterprise (NHO) [7]. AI can automate repetitive and time-consuming tasks in the real estate market, such as writing image captions. Currently, over 3 million images on Finn.no are manually captioned each year, a process that is not only labor-intensive but also prone to inconsistencies, repetitiveness and spelling errors [3].

Automating this process with a fine-tuned image captioning model can significantly streamline the creation of real estate listings. This model can ensure that all images receive high-quality, consistent captions quickly and efficiently. Beyond this, the model can extract structured data from images, enhancing the precision of Solgt.no’s Automated Valuation Models (AVM) used for property value estimation. This improves the efficiency of real estate transactions and provides more accurate market insights.

Developing this image captioning model aims to revolutionize Norway’s real estate industry, making it more efficient and innovative. This project will demonstrate how a targeted AI solution can address specific industry needs, leading to broader applications and advancements in Norwegian AI research and development.

1.3 Prerequisites

This master’s thesis does not build upon any previous work. The starting point is a raw dataset of images with captions written by real estate agents, and no further research or processing on the dataset has been done beforehand. The aim is to use the dataset together with available software and hardware to obtain a model that achieves the objectives of this project.

1.3.1 Hardware

GPU Server: Graphics Processing Unit (GPU) is needed to train an image captioning model. The GPU used for this project is Solgt.no’s server, which is equipped with the NVIDIA GeForce RTX 4090 graphics card. The server is running on driver version 545.23.08 and CUDA version 12.3. CUDA is a parallel computing platform and application programming interface model created by NVIDIA, which allows software developers to use a CUDA-enabled GPU for general-purpose processing. The GPU is available through Solgt.no’s virtual private network (VPN).

CPU: NTNU provided a Dell Optiplex computer equipped with an Intel(R) Core(TM) i5-10500 Central Processing Unit (CPU) operating at a base frequency of 3.10 GHz. The

computer has a total of 32.0 GB of random-access memory (RAM). A subgoal is to find a neural network where this available CPU is sufficient to run the fine-tuned model.

1.3.2 Software

Transformers with Python: The Hugging Face’s Transformers library is used along with PyTorch. Additionally, PIL is used to handle images. It is compiled in Jupyter Notebook on Solgt.no’s VPN using Jupyter Lab.

React app: For the React app used for user test, python with Flask for handling API requests is used for the backend. CSS and JavaScript in the frontend. Axios is used to manage HTTP requests between React and Flask. Node.js and npm help manage frontend dependencies and build tools. This combination is used to make a full-stack application to create a user interface for model inference.

1.4 Report Structure

The report structure begins with an introduction to the theory behind the deep neural networks utilized and other state-of-the-art models. Following this, the report details the methods for evaluating language generation models, establishing a foundation for understanding the project’s evaluation framework, which includes automatic and human evaluation. The report introduces machine learning applications in the Norwegian real estate industry, highlighting platforms like Finn.no and Prosper AI and their relevance to the project’s objectives.

The data chapter gives an overview of the dataset used in the project, detailing the data format, preprocessing steps, and the creation of final datasets. This sets the stage for the chapter on the methodology, which describes the training, implementation, and evaluation processes of the BLIP model of different sizes.

The results chapter presents the outcomes of the training and evaluation, including comparisons between different models, survey results, and user test results. This is followed by a discussion chapter that analyzes the dataset properties, preprocessing comments, and evaluation of the best-performing models and addresses the limitations of the study. Finally, the conclusion summarizes the findings and suggests potential directions for future work.

The appendices include additional resources examples of generated captions, demonstration videos of the website used for user test, and code snippets providing a foundation for further exploration and understanding of the project’s scope and results.

Preliminaries

This chapter covers the theory behind the methods used in this master's thesis. Furthermore, it introduces the state-of-the-art models within Large Language Models and Generative AI. Lastly, including related work with machine learning in the Norwegian Real Estate industry.

2.1 A Simple Introduction to Neural Networks

Each neural network (NN) is composed of layers of nodes or artificial neurons, including an input layer, hidden layers, and an output layer. These nodes are the computational units in a network, and an activation function decides the computation. Each node connects with others and has its weight and threshold. If a specific node's output exceeds the threshold value, it's activated and sends data to the next layer of the network [8]. An activation function is therefore used to determine which of the neurons in a layer that should be activated for a specific sequence in a learning iteration. A criterion for an activation function is that it must be differentiable [9].

Backpropagation is commonly used when training a neural network. It is a way for the network to compare the prediction and the ground truth and adjust the weights thereafter. It is a method similar to the way we humans correct ourselves and improve our decisions with feedback. During backpropagation, gradients are computed using the derivative of the activation functions to decide if the node should be activated. This is essential for updating the weights and biases in the network [8]. Since the network adjusts its weights using the ground truth, it is important that the true data is correct. It is, therefore, vital to use a good dataset that is of sufficient quality.

Neural networks require training data that is balanced. Once they are trained, they become valuable tools in computer science and artificial intelligence (AI). They are capable of classifying and clustering data, enabling tasks such as speech recognition or image recognition to take minutes instead of hours compared to manual identification by humans [8]. To evaluate the performance and generalization of the model, one needs to test the model on unseen data. Depending on the size of the dataset, a common practice is to hold out 10 to 20 percent of the total set to use for testing. It is critical to avoid data leakage, meaning this data must not be used before evaluating the model [10].

Regularization is a common method to prevent model overfitting, which is when a model

learns the patterns on the training dataset too well and performs poorly when validation loss increases while training loss decreases. Causes of overfitting vary, but several solutions exist. L1 and L2 norm regularization reduces loss by penalizing absolute (L1) or squared (L2) weight values. Dropout, which randomly deactivates neurons during training, also helps prevent overfitting. Other techniques include data augmentation and early stopping [11, 10].

The neural network architectures are getting larger and more complicated, and they require huge datasets to be able to train these models. *Transfer learning* has therefore become a standard method for many AI tasks. It is when one uses a model that is pre-trained on a large dataset and then customized to a given task using a smaller task-specific dataset [12]. During transfer learning, the model's weights and biases are adjusted. These adjustments are saved, so when one wants to use a fine-tuned model, then the updated weights must be employed.

2.2 Image Captioning

Developing image captioning in the context of machine learning (ML) is to train a model to predict a caption in a desired language for an image. The steps of doing this are first to find a suited neural network to train, then fine-tune this on a selected dataset and use this model for inference [13]. This task appears straightforward, yet it can be intricate and sometimes not even feasible with the available tools such as dataset and GPU.

Image captioning was considered one of the most complicated generative tasks within machine learning until multimodal models were developed. Multimodality is a machine learning system that can combine multiple types of data such as text and images [14]. Designing a model with both understanding-based tasks and generation-based tasks requires complicated architecture. Early attempts at image captioning primarily relied on computer vision techniques yet struggled to capture the nuanced semantics and relationships within images. Adding natural language processing to image captioning was a big change that led to making stronger and more aware models. Multimodal language models, like OpenAI's Generative Pre-trained Transformer (GPT), were inspired by big language models and used both pictures and text to understand things better. These models learn from large sets of paired pictures and captions, which helps them understand how visual features and words relate to each other. Multimodal language models usually consist of two main parts: a vision module and a language module. Currently, the most popular multimodal models use *transformers* for both of the modules [15].

2.2.1 Foundation Models

Today's most publicly used AI models are *foundation models*. These are a class of large-scale machine learning models pre-trained on extensive datasets to capture a wide range of knowledge and skills [16]. Examples of these models include the GPT-4 and Google Gemini, which serve as a foundational base for developing specialized applications with high precision. They are capable of performing a variety of tasks across different domains, including natural language processing (NLP), image recognition, and generative tasks like text and image creation. What sets foundation models apart is their ability to adapt to new tasks with little additional training, leveraging their generalist pre-training to provide high-quality results on tasks they were not explicitly trained to perform. This adaptability

and broad applicability make them powerful tools for advancing AI and developing new applications across industries. The foundation models can be adapted to perform more specific tasks with higher precision. Image captioning constitutes one category of adapted foundation models that unifies vision and language tasks [16].

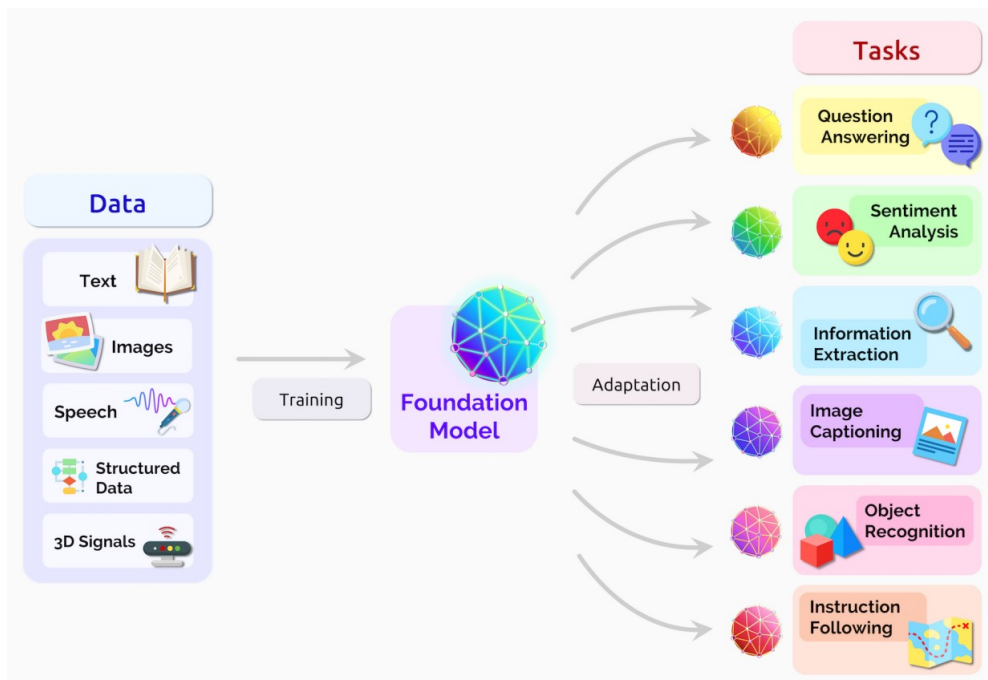


Figure 2.2.1: Illustration of different adaptations of foundation models[16]

2.3 Transformers

To understand the basics of these popular new models, one must be familiar with the concept of transformers, as it is the foundation of many state-of-the-art models. Transformers were introduced in 2017 in the paper "Attention Is All You Need" [17]. The uniqueness of transformers is because of the three following key concepts.

- **Self-Attention:** This architecture is composed of an encoder and a decoder, each consisting of multiple layers. The model leverages self-attention mechanisms that help it to process input sequences in parallel, significantly increasing efficiency and reducing training time.
- **Multi-Head Attention:** This mechanism allows the model to weigh the importance of different words in the sequence, regardless of their positions, for a given task. By computing a representation of the sequence that captures these dependencies, it handles long-range interactions between words effectively.
- **Positional Encoding:** Since the model does not inherently consider the order of words in the input, positional encodings are added to give the model access to the position of words within the sequence. This helps maintain the sequence order necessary for tasks like translation.

To contextualize, transformers use encoders and decoders. The encoder vectorizes a sequence of tokens. The tokens are a numerical representation of the input and are typically a lower-dimensional representation of the input, which captures the most important information. For instance, an image of a bright kitchen or the sentence "The kitchen is bright" is converted into tokens. These tokens are then refined using the multi-head attention mechanism, which highlights keywords. In the sentence example, "kitchen" and "bright" would be emphasized as important tokens. The decoder's task is to take this encoding and generate the output. The decoder uses multiple layers of neural networks to process the encoded data and produce the final output, like a translated sentence or an image caption. It also employs an attention mechanism that enables the model to concentrate on specific sections of the encoded data while generating the output [18].

Before the introduction of Transformers, Recurrent Neural Network (RNN) was commonly used for text processing. Versions of RNNs faced challenges with large sequences as they tended to forget earlier information by the time they reached the end of a sequence, often described as short short-term memory. RNNs are also hard to train because they process words sequentially, and it is therefore not possible to improve or speed up the process with more graphics processing units [19]. Transformers, on the other hand, have no recurrent units and therefore need less memory and require shorter training time. With transformers, one can parallelize the training, which enables the possibility of training models on a really large amount of data. Enabling faster training on larger datasets, emphasizing the importance of attention mechanisms in achieving high performance in natural language processing tasks [15].

2.3.1 Vision Transformers

A vision transformer (ViT) is an alternative to a Convolutional Neural Network (CNN) in computer vision tasks. ViT has proven to be an algorithm suited for larger scaling and is now the standard for image-to-text tasks. A standard approach is to pre-train ViT on a large text corpus, then fine-tune on a smaller task-specific dataset for the ViT to fulfill its potential [20]. A GPU-power efficient method is to pre-train on images with lower resolution and increase the image resolution for fine-tuning to smaller downstream tasks. In this way, less computation is needed for the general model, and task-specific is still able to capture details.

Model Architecture: The ViT model follows a straightforward adaptation of the transformer architecture used in NLP. An input image is divided into fixed-size patches, each of which is linearly embedded into a vector using *tokenizers*. These patch embeddings are then augmented with positional embeddings to retain spatial information, and an extra learnable class embedding is appended to the sequence. This sequence of embeddings is then fed into a standard transformer encoder, which consists of multiple layers of multi-headed self-attention and feed-forward neural networks [20]. The architecture of the ViT can be summarized as follows:

- **Image Patching:** The image is split into patches of a fixed size, typically 16x16 pixels.
- **Linear Embedding:** Each patch is flattened and projected into a vector of a specified dimension.

- **Positional Embedding:** Learnable positional embeddings are added to the patch embeddings.
- **Transformer Encoder:** The sequence of embedded patches, along with a class token, is processed through the transformer layers.
- **Classification Head:** The output corresponding to the class token is passed through a multi-layer perceptron (MLP) for the final classification.

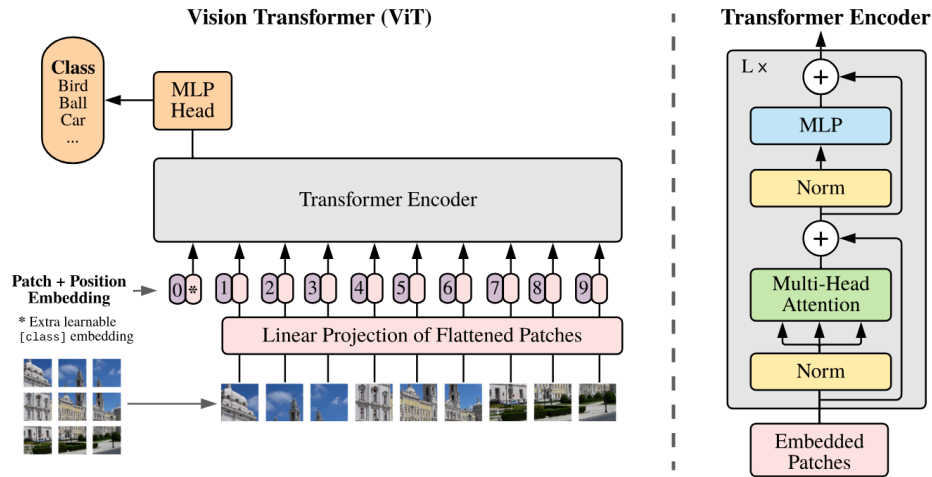


Figure 2.3.1: Figure illustrating a ViT [20]

In comparing CNN and Vision Transformer models, there are notable differences in size, memory needs, accuracy, and performance. CNNs are compact and efficient, ideal for constrained environments, and perform well across various image processing tasks. Vision Transformers, however, excel at capturing global dependencies in images, which is especially beneficial for image captioning as they can focus on different parts of an image to generate accurate and contextually relevant captions. Yet, ViTs are generally larger and more memory-intensive, making them less suited for resource-limited settings. The choice between CNNs and ViTs depends on the specific needs of the task, considering the available resources and the balance between model complexity and performance. As advancements continue, both architectures are expected to evolve, offering more tailored options for different applications [21].

2.4 Convolutional Neural Network

Convolutional Neural Network (CNN) is a type of deep learning architecture specifically designed to process and analyze visual data such as images. While different applications of CNN may require specific architectural modifications, they are all built on the same fundamental principles.

2.4.1 CNN Architecture

A CNN is composed of layers of neurons. A neuron is a basic computational unit that takes inputs, processes them by conducting a weighted sum followed by applying an activation function, and then produces an output. Each neuron in a CNN is designed to identify specific

features or patterns in the input data. A layer is a group of neurons arranged in a specific sequence. Each layer performs different operations on its inputs. A typical CNN consists of three types of layers between the input and output layers, which are Convolution, Pooling, and Fully Connected (Dense) layers. The Convolution layer uses various filters to scan and transform the input image, extracting important features, while the Pooling layer reduces the size of feature maps to enhance efficiency and reduce computational load, and the Fully Connected layer utilizes the processed information to classify the input into categories [22].

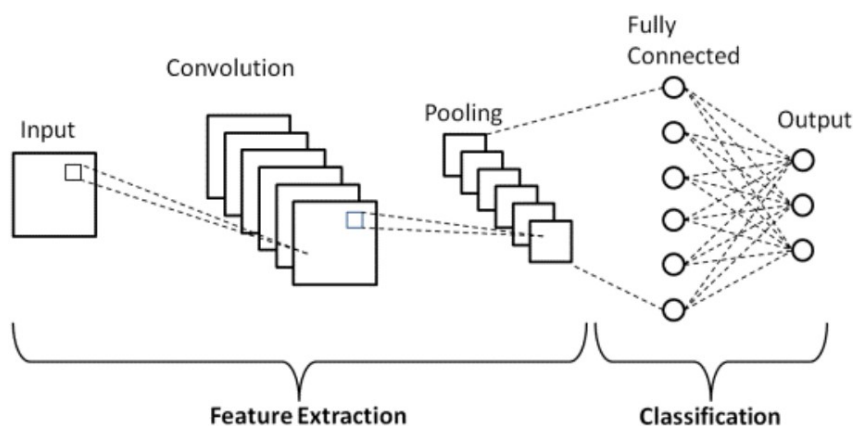


Figure 2.4.1: Simple illustration of a CNN [23]

2.5 State-of-the-art NLG

The field of Natural Language Generation is constantly evolving, making it challenging to stay up-to-date with the latest advancements. The following subsections will introduce Hugging Face and some high-performing state-of-the-art models.

2.5.1 Hugging Face

There is a division in the field of machine learning development between open and private source models. Over the past years, private source models have been the best performers, but open-source models are catching up. Thanks to the open-source community platform Hugging Face, many developers now have a good overview and easy access to the most popular and recent developments in the field. Hugging Face has created a transformers library that makes it easier to develop and use transformers through the library. Hugging Face contains thousands of open-source pre-trained models, datasets, and unique architectures [24].

One can find a leaderboard on the best performing Large Language Models on LMSYS Chatbot Arena leaderboard [25]. This specifically compares OpenAI’s private sourced GPT models with open-source alternatives on a set of benchmarks. Last updated, 2024-05-27, GPT-4o was number one on the leaderboard out of 102 models in total and 1,149,962 crowdsourced votes [25].

2.5.2 GPT-4 by OpenAI

GPT-4 is a large foundation model, whereas GPT-4V (Vision) is an integrated adaptive multimodal transformer-based model. This means that GPT-4 is able to do image-to-text tasks. GPT-4 has become the benchmark within image-to-text models, being superior in terms of results. It supports most of the world’s languages. There are, however, some downsides with GPT-4. Firstly, it is a private source algorithm with payment requirements. It is therefore expensive to use their API when it is frequently used for the same repetitive task as the one in this project. Furthermore, the size of the model is unknown, and it does not support fine-tuning. Therefore, using this large and dynamic model for rigid and repetitive tasks is inefficient. GPT-4 as a foundation model is said to be of more than a trillion parameters, and its energy demand is not sustainable [26].

2.5.3 NoraLLM

NoraLLM is a family of large Norwegian language models developed collaboratively by the University of Oslo, the National Library of Norway, and other institutions. It utilizes the digitalization of the Norwegian National Library to pre-train on Norwegian text, resulting in over 7 billion parameters. These models are designed for text generation tasks that include web data and code. NoraLLM is under constant development and aims to enhance Norwegian natural language processing capabilities and especially improve knowledge within this field [27].

NoraLLM versions are trained with GPU using the supercomputer LUMI. It focuses on having a small carbon footprint by being one of the most eco-efficient data centers globally. It covers 100% of its energy consumption with renewable electricity [27, 28].

2.5.4 Stable Diffusion 3 by Stability AI

Stable Diffusion 3 (SD3) is considered to be one of the most prominent foundation models for text-to-image tasks. It has a multimodal diffusion transformer architecture and has proven to achieve high quality from text to image generation. Stability AI has shared its source code as well as model weights for Stable Diffusion on GitHub [29].

2.5.5 BLIP by Salesforce

Bootstrapping Language-Image Pre-Training (BLIP) is a popular state-the-art image captioning model. There are two main perspectives when using Vision-Language Pre-training (VLP) that make BLIP a popular open-source model for pre-training and transfer learning:

- **Model perspective:** BLIP has a Multimodal mixture of Encoder-Decoder (MED) architecture, which separates it from its ancestors that are either encoder-based or encoder-decoder models. The encoder-base models are hard to transfer for text generation tasks, and encoder-decoder models are worse for image retrieval tasks. With MED, the model is much better for pre-training and transfer learning. This multi-task approach ensures robust learning of both vision and language representations [30].
- **Data perspective:** When scaling up the dataset, BLIP has been pre-trained on data collected from the web with noisy text, which is suboptimal for VLP. It is, therefore, important that the datasets used for fine-tuning are of high quality. BLIP uses a

bootstrapping method called CapFit that creates synthetic image captions given web images, and a filter to remove noisy captions. This was created to enhance the pre-training using data collected from the web, making it more robust towards noise [30].

The image resolution used during pre-training is 224x224, and during fine-tuning, it is 384 x 384. It is for more energy-efficient training by reducing the GPU demand with lower image resolution.

2.5.5.1 Loss Calculation for BLIP

BLIP employs a combination of three loss functions to effectively train a multimodal model capable of both vision-language understanding and generation tasks. These combined loss functions enable BLIP to robustly learn from both noisy and clean image-text pairs, ensuring strong performance across various vision-language tasks [30].

- Image-Text Contrastive Loss (ITC)
- Image-Text Matching Loss (ITM)
- Language Modeling Loss (LM)

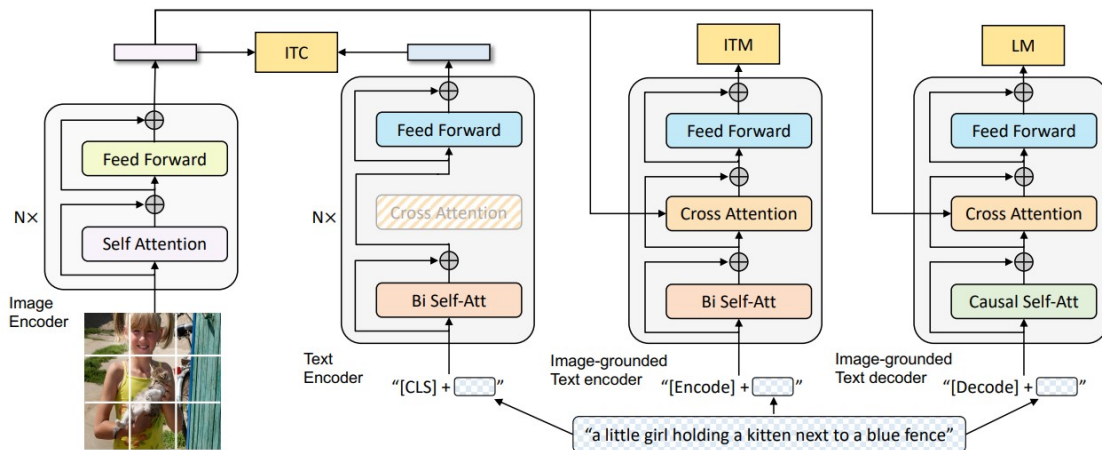


Figure 2.5.1: Architecture of BLIP [30]

ITC aligns the feature spaces of the visual and text transformers by encouraging matching image-text pairs to have similar representations compared to unmatched pairs. This is achieved using a momentum encoder to generate features and soft labels, accounting for potential matches within the unmatched pairs.

The second loss function is ITM, which focuses on learning a multimodal representation that captures the fine-grained alignment between vision and language. This is a binary classification task where the model predicts whether an image-text pair is matched or unmatched, utilizing an ITM head.

The third loss function is LM, which optimizes the model to generate textual descriptions given an image. This loss function employs a cross-entropy loss to maximize the likelihood of the text. Cross-entropy involves taking the negative logarithm of the confidence score

and using it to adjust the bias and weights. This method penalizes values that are far from the correct prediction, as illustrated in figure 2.5.2. A predicted probability of 1 incurs no loss, while a confidence score of 0 results in infinitely high loss [10, 31].

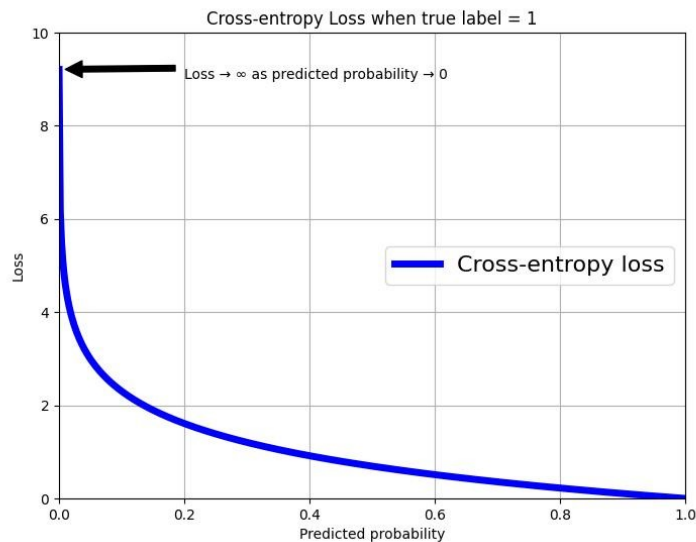


Figure 2.5.2: The cross-entropy loss function illustrated

2.6 Evaluation Metrics for Text Generation

Evaluating the performance in natural language generation (NLG) are often difficult because the tasks are usually open-ended [32]. Humans would most likely come up with different variations for different sentences explaining the same thing, and there is usually never a universally correct answer. One has to find metrics that are suited for the tasks. This project is focusing on two main types of evaluation.

- **Automatic Evaluation**

- Language quality and diversity evaluation
- Precision evaluation using ground truth

- **Human evaluation**

The automatic evaluation methods can be done efficiently for each iteration of tuning and adjustment of the dataset, and human evaluation is especially important in the finishing stages.

2.6.1 Automatic Evaluation

To gain an overall evaluation one can combine different metrics to evaluate the language quality. TTR and Yule’s K are to evaluate the diversity of the vocabulary, and BLEU score and Semantic Textual Similarity is to calculate sentence similarity between generated text and ground truth. In this case the ground truth is the realtor-written caption for the given image.

2.6.1.1 BLEU

Bilingual Evaluation Understudy (BLEU) NLP is a method used to assess the quality based on how closely a machine’s translation resembles a professional human translation. The core principle of BLEU is that the closer a machine translation is to a professional human translation, the better it is [33]. Developed at IBM in 2001, BLEU was among the earliest metrics to demonstrate a strong correlation with human evaluations of translation quality and continues to be one of the most widely used automated and cost-effective metrics [34].

The output of BLEU is a score between 0 and 1, where 1 is closest to the target sentence. It is mainly suited for translation tasks instead of open-ended tasks such as image captioning. The issue with the BLEU score is that it directly compares the generated and the true text using n-gram Co-occurrence. An n-gram is a sequence of n words from a given text. BLEU computes precision for n-grams of different lengths. For each length, the precision is calculated as the number of matching n-grams divided by the total number of n-grams in the generated text, referred to as the candidate. This measures the extent to which the candidate text contains n-grams that are also present in the target texts [35]. The n-gram precision p_n is calculated as:

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n\text{-gram} \in C} \text{Count}_{\text{clip}}(n\text{-gram})}{\sum_{C \in \{Candidates\}} \sum_{n\text{-gram} \in C} \text{Count}(n\text{-gram})} \quad (2.1)$$

where $\text{Count}_{\text{clip}}(n\text{-gram})$ is the number of n-grams in both the candidate and the references, clipped to the maximum count in the references.

To avoid favoring shorter texts, BLEU includes a brevity penalty.

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases} \quad (2.2)$$

Where c is the length of the candidate text, and r is the reference length .

The BLEU score combines the precision scores with the brevity penalty:

$$\text{BLEU} = BP \cdot \exp \left(w_n \sum_{n=1}^N \log p_n \right) \quad (2.3)$$

where N is typically 4, and w_n are the weights for each n-gram precision, often set to $\frac{1}{N}$ [35, 36].

BLEU provides a quantitative measure of text similarity by evaluating n-gram overlaps and adjusting for text length, making it a robust tool for text evaluation tasks. In summary, BLEU evaluates text similarity by assessing the precision of n-gram overlaps between the candidate and reference texts.

2.6.1.2 Semantic Textual Similarity

The Semantic Textual Similarity (STS) process is used to determine how similar the captions generated by a system are to the true captions. It uses a Sentence Transformer model to vectorize generated captions and the true captions. It then measures how close the vectors of the generated captions are to the vectors of the true captions with cosine similarity

[37]. The closer the vectors are in direction, the more similar their meanings. It is another metric to understand how accurately the system is generating captions compared to what is expected, providing a straightforward way to evaluate the system's performance in generating meaningful captions.

If the fixed vector size is very large, then this is a very computationally demanding task. Luckily, for this project, the captions are one or two sentences, and have limited vocabulary since it is only fine-tuned on real estate related vocabulary. It should therefore be a suited measurement for this task.

Another upside of using STS in addition to BLEU is the benefit that similar words can be close to each other in a vector space. When the formulation and use of words are not exactly the same, but the content is similar, then vectors can be similar as well. E.g. The famous English paragram "*The quick brown fox jumps over the lazy dog.*" and the sentence "*A fast, brown-colored fox leaps over a sleepy canine.*" [38]. These two sentences has very similar meanings. The words "quick" and "fast," "jumps" and "leaps," "lazy" and "sleepy," "dog" and "canine" are synonyms or very close in meaning. Therefore, the STS score between these sentences would be high. On the other hand, the BLEU score measures the n-gram Co-occurrence between the sentences. Since these two sentences have few exact n-gram matches other than "the", "fox" and "over", the BLEU score would be low. The BLEU score relies on precise wording and structure, which differs in these sentences despite their semantic similarity.

2.6.1.3 Average Sentence Length

Average sentence length can indicate a caption's complexity. Short sentences are usually simpler and less detailed. By investigating the sentence length, one might be able to understand the average complexity of the sentences.

2.6.1.4 TTR

The Type-Token Ratio (TTR) measures lexical diversity, which indicates how varied the vocabulary in a text is. Tokens represent the total number of words in a text, while types refer to the number of unique words [39]. A higher TTR score suggests greater lexical diversity. The formula for TTR is:

$$\text{TTR} = \frac{\text{Number of Types}}{\text{Number of Tokens}} \quad (2.4)$$

2.6.1.5 Yule's K

Yule's K is a more complex measure of lexical diversity compared to TTR. Yule's K quantifies lexical richness by calculating the ratio between different unique word types and the total number of word tokens in the text [40]. Unlike TTR, Yule's K inversely correlates with lexical richness. A higher Yule's K value indicates lower vocabulary richness. It indicates that the text has a high level of repetitiveness, often utilizing a smaller core vocabulary. On the other hand, a low value suggests a linguistically diverse text with less repetition of individual words [41].

Yule’s K is mostly used in linguistic studies where textual complexity and vocabulary diversity need to be compared across various texts or authors. It is not commonly known for being a metric for NLG tasks, but it can be a good addition to the overall evaluation. Additionally, Yule’s K is sensitive to the length of the text. Larger texts might show greater diversity simply due to the increased opportunity to use different words. However, the generated captions in this task are limited to one or two sentences, which constitutes the decision to use Yule’s K as an additional metric for this NLG task [41]. The Yule’s K measure is calculated using the following formula:

$$K = 10^4 \left(\frac{-1}{N} + \sum_{m=1}^{m_{\max}} \frac{V(m, N) \cdot (m/N)^2}{S_1^2} \right) \quad (2.5)$$

Where:

- N is the total number of tokens.
- $V(m, N)$ represents the number of words that appear exactly m times in the text.
- m_{\max} is the maximum frequency of any word in the text.
- $S_1 = \sum_{m=1}^{m_{\max}} m \cdot V(m, N)$, which is the total number of words in the text

2.6.2 Human Evaluation

Human evaluation plays a crucial role in assessing natural language generation. The aim of these machine learning tasks is to train a machine to generate language that mimics human speech without being identified as a machine [42]. While the language may be grammatically correct and perform well according to automated evaluation metrics, it can still lack the authentic feel of human language. This is often referred to as the uncanny valley within ML [43]. Another challenge with using machine learning for text is hallucination. It is when natural language generation models generate false output text. The issue with hallucinations is that they might go undiscovered because the model writes them with such confidence. It is not uncommon that they generate text that is nonsensical or inaccurate [44]. It is therefore crucial to include human interaction in the evaluation process of NLG models.

A natural human interaction is when the developer evaluates the performance of the model throughout the development. **Developer’s evaluation** ensures that the model meets the desired standards and functions correctly. Furthermore, human evaluation is essential as it offers insights into the model’s capacity to produce coherent, relevant, and contextually appropriate responses, and to identify hallucinations and avoid the uncanny valley. By evaluating the test results throughout the tuning process, one is able to identify and address any issues early on, leading to a more reliable and effective NLG model.

However, this can cause a developer bias, which is the tendency of developers to make a product with its own ideal environment as the goal. The performance should therefore, also be evaluated by neutral human evaluators. The Turing test is the most famous method for this type of evaluation, but simpler methods such as surveys or crowdsourced user tests are also suitable for tasks like image captioning [45]. These approaches provide unbiased feedback and ensure the model’s effectiveness in real-world scenarios.

A survey is a beneficial method to evaluate the results for multiple reasons. Surveys allow for collecting diverse feedback from a wide range of users, resulting in a more comprehensive assessment of the model’s performance. They can include specific questions about the relevance, coherence, and accuracy of the generated content, helping to identify strengths and areas for improvement. Additionally, surveys can capture subjective preferences, offering valuable insights that automatic metrics might miss. By analyzing survey results, developers can gain a clearer understanding of how well the model performs in practical applications and make informed adjustments to enhance its overall effectiveness.

User testing the models on the target audience is important because it provides direct insights into how the development performs in real-world scenarios. It allows developers to observe how actual users interact with the generated content, highlighting any usability issues or misunderstandings. Furthermore, the feedback from users can reduce the developer’s bias since it incorporates the perspectives and experiences of the respondents. This approach ensures that the model meets the needs and expectations of its intended audience, leading to a more user-centered and effective NLG model.

2.7 ML for Norwegian Real Estate Listings

There are currently not many publicly known AI products that are similar to this project that are used in the Norwegian real estate industry. However, there are a few AI developments in the industry of real estate listings introduced below.

2.7.1 Finn.no

A developer at Finn.no agreed to do an interview about AI within their company. Note that this interview was done over e-mail and in Norwegian, so the answers are translated into English.

2.7.1.1 Floor Plan Classification

At the end of 2023, Finn.no launched an automated floor plan classification model [46]. The goal was to make the floor plan easily accessible since they noticed that their users were looking for the floor plan in their advertisements.

2.7.1.2 BLINK - Text Generation

BLINK is an audience-targeting marketing tool, that promotes real estate advertisements on other platforms based on your search history on Finn.no. The ads that are placed on other platforms have AI-generated titles. The headlines are generated by an AI that has been trained to understand what is important in real estate listings [46]. The interview object said in her interview that they did not use any Norwegian LLM to generate titles, but instead, they use OpenAi’s API. The input is information from the advertisement as props, and the generated title is based solely on text and not images. To evaluate their model, they measure the efficiency of the titles with click-through rate (CTR). It is a measurement that counts how many people who click on an ad compared to how many who watch the ad. By comparing the CTR between the AI-generated captions and the customer-written ones, they find out if the AI-generated ones deliver more clicks on the ad.

2.7.2 Room Classification by Hjem.no

Hjem.no is a competitor to Finn.no, and they launched their website at 11.04.2024. This is another marketplace for buying, selling, and renting homes, cabins, recreational properties, and commercial and office spaces [47]. They automatically classify bathrooms and kitchens in addition to floor plans.

2.7.3 Automatic Generated Prospects by Prosper AI

Prosper AI has been contacted for the purpose of this master's thesis, but they did not comment on the launch date of this product or the product design.

In March 2024, Shifter.no announced that Prosper AI, is going to make a prospect generator for Norwegian real estate advertisements [48]. The product is an AI back-office for prospects and image captions for real estate advertisements. According to Prosper AI, it is the first advanced, Norwegian-developed service that functions as a digital back-office product without compromising the quality of using AI. The goal is to reduce hours of work to 15-30 minutes [49].

Prosper AI proves that there is market interest in this type of product. The recent engagement and substantial investment by a prominent company, such as Computas, in developing a product similar to this thesis project underscores the market relevance and potential impact of the innovation proposed in this research [48]. The attention and resources directed toward a similar product confirm the practical viability of the concept but also emphasize its timeliness and significance within the industry. However, the difference is that this thesis' goal is to contribute to research on the Norwegian generative model by being one of the first public projects to do this.

Chapter 3

Data

This chapter introduce the dataset that has been used in this project. It includes detailed information on the data format, the process of data cleaning, and analyzing the data for features that can impact the performance of the models. This covers the required preprocessing steps that need to be done before training a model on this data, which is vital work for the project.

3.1 Dataset

The dataset used for this thesis is based on 10.000 listings in the Oslo area. It consists of approximately 300.000 images with corresponding captions written by realtors. This is a dataset that contains all sorts of images and captions that are used in real estate advertisements. This can be images of everything from kitchens, living rooms, and bathrooms to facades and images of the local area.

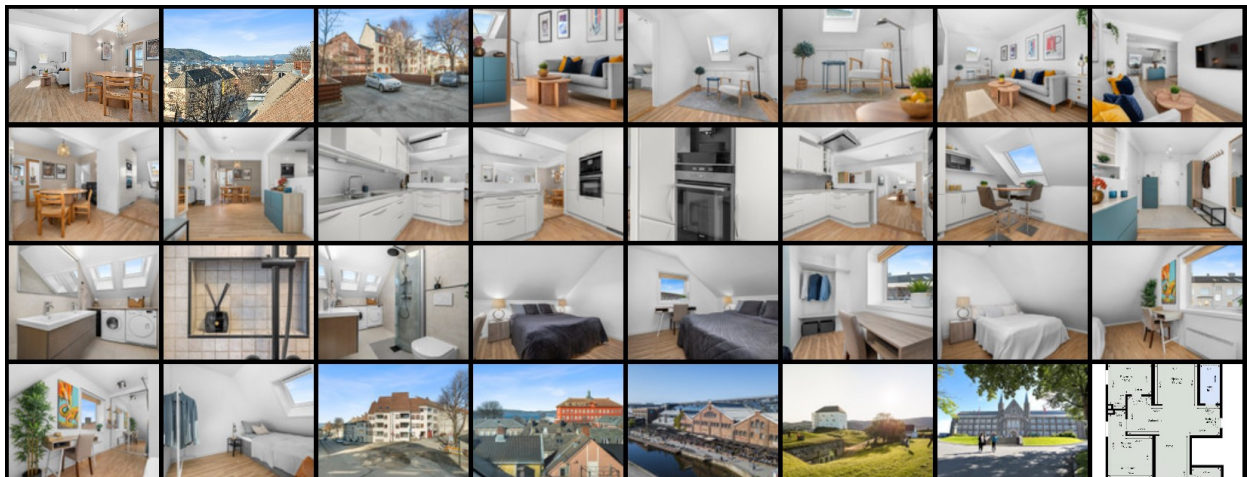


Figure 3.1.1: Screenshot of an image gallery from an advertisement on Finn.no [50]

3.1.1 Data Format

The goal is to have a data format which is a suitable representation of the text belonging to its image. Requirements for the data format are that it can be used in preprocessing stages and as input in different neural networks.



Figure 3.1.2: Example of an image with caption from Finn.no [50]



Figure 3.1.3: Example of an image with caption from Finn.no [50]

3.1.1.1 COCO

The dataset is in Common Object In Context (COCO) format, which consists of 2 parts. COCO-formated data is commonly used for training, validation, and testing for many state-of-the-art object and detection models. Firstly, we have all the images stored as jpg files, and secondly, the annotations are stored in JavaScript Object Notation (JSON) format [51]. The following is how the JSON-object in this dataset is structured.

The **images** section is an array with each entry representing an individual image in the dataset. Key details about each image are provided as follows.

```
{
  'width': width in pixels,
  'height': height in pixels,
  'id': unique image ID,
  'file_name': "relative path to the image file"
}
```

The **annotations** array contains detailed information about objects within each image. Each annotation includes this.

```
{
  'image_id': "Image associated with this annotation.",
  'Description': "The caption corresponding to the image."
}
```

3.1.1.2 Arrow

After the preprocessing stages, the data is converted into arrow files. Arrow is beneficial when one wants to process and quickly move a large amount of data without serialization overhead. It supports different programming languages and allows most programming tools and libraries. Apache dataset supports arrow data, which is the desired input for most transformers [52].

3.1.1.3 RGB images

The Red, Green, and Blue (RGB) color space is the standard for computer vision tasks. Since all the images in the dataset are initially jpg files, they need to be uncompressed into RGB images before being used in the neural network [53].

3.2 Preprocess

It is vital to have a dataset that is large, diverse, balanced, and clean to train a high-performing model. Composing an adequate dataset can sometimes be the most challenging task within machine learning. Preprocessing the data is therefore an essential part of training a model to generate accurate captions.

This thesis focuses on making a high-quality **dataset for kitchens** as a proof of concept. If it is possible to structure a dataset with images and captions of kitchens that are used to fine-tune a high-performing image captioning model, then it should be possible for all types of rooms. The reasoning behind choosing kitchens is that captions of kitchens can contain many details, no details at all, and everything in between. When training on the dataset without any preprocessing, BLIP struggled the most with kitchens and was not able to

generate anything other than one or two of the exact same sentences each time it detected a kitchen. This poor performance is illustrated in Section 5.2.1. If the model generates precise captions for kitchens, then it should be feasible to do the same for the rest of the rooms. Images of floor plans, living rooms, bedrooms, and entries were much better, even without a dataset preprocessing.

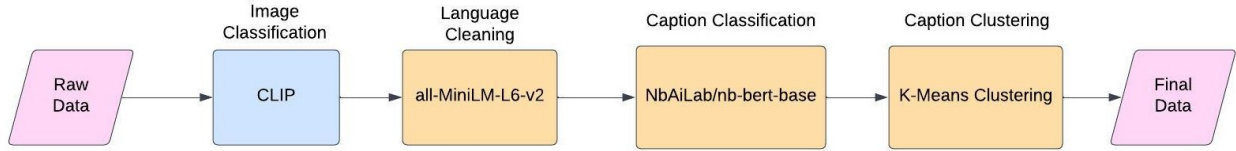


Figure 3.2.1: Flowchart visualizing the preprocessing stages, with the blue color being images as input and yellow as text input.

3.2.1 Classify Images

The first step in composing the dataset is to classify all images into Bedroom, Bathroom, Kitchen, Living room, Floor plan, Garden, Facade, Entry/Hallway, Garage, and Others. There are different approaches to classifying images. Using multi-class CNN would be a natural approach if one wants a highly accurate classifier. However, that requires a labeled dataset with the desired classes, hyperparameter tuning, and the use of GPU. Instead, one can use a ViT-based approach with Contrastive Language-Image Pre-Training (CLIP). CLIP can classify images based on textual descriptions. It compares unseen images with text-described classes. As a result, image classification is performed without requiring labeled training data for specific classes [14]. The defined classes are as follows.

Category Name
Balcony
Bathroom
Bedroom
Entry
Facade
Floor Plan
Garage
Garden
Kitchen
Living Room
Others
Storage Room

Table 3.2.1: Image Categories

Classifying the whole dataset of 300.000 images took approximately 55 hours running on a CPU. CLIP proved itself to be sufficiently accurate without further tuning.

3.2.2 Caption Cleaning

After having classified images into categories, the kitchen images are then further filtered between high-quality and low-quality captions. This is done with the sentence transformer *all-MiniLM-L6-v2* [54]. A manually labeled dataset was prepared for this cleaning process, including bad sentences with spelling errors or content unrelated to kitchens. The sentence transformer was fine-tuned on this dataset. Captions are then classified as either high or low quality based on a threshold of 50 percent similarity to the low-quality sentences. The images with captions of low quality are then discarded.

To enhance the dataset even further, kitchen-specific names and years in the captions are replaced with placeholders "XXX" and "NNNN." Captions often include names or years, such as "Fint IKEA-kjøkken fra 2019." This specific information is not useful for the training process, so instead of discarding these captions, the year is replaced with "NNNN" and the kitchen supplier name with "XXX." This allows the model to generate captions that can still include this type of information. For example, the caption "Fint IKEA-kjøkken fra 2019" becomes "Fint XXX-kjøkken fra NNNN" in the dataset.

3.2.3 Classify Captions

Furthermore, the dataset is classified into four different categories. Descriptive, Informative, Creative, and Irrelevant. It is done with the transformer, *NbAiLab/nb-bert-base*, which is a model trained on digital documents from the Norwegian National Library [55]. The model is primarily designed for tasks like token classification and sequence classification. It is based on the BERT bidirectional transformer architecture, which is good at understanding context but does not generate text [55]. This is tuned with a training set consisting of 2000 captions that are manually classified into the four categories.

When deciding the dataset, one has to select how large a percentage is desired from each category. For example, having just descriptive captions in the training set reduces the possibility of the model generating words that are purely descriptions of the room. For example, "white cabinets" is descriptive, but "spacious and bright kitchen" also describes the kitchen, but it is based on how the room feels and is therefore in the creative category. Finding a good balance between the categories can lead to a more lively language generated by the model, e.g., "A spacious and bright kitchen with white cabinets." Including these descriptions of how the room feels and other adjectives is one of the most distinct differences between a standard NLG task since the goal is to inherit how realtors write their captions today, not just describe the image straight forward.

3.2.4 Balance The Data with K-Means Clustering

There is still an imbalance in the types of captions within each class. For example, there are many more "Kjøkken med integrerte hvitevarer" and "Kjøkkendetalj" captions than "XXX kjøkkeninnredning med glatte hvite skapfronter i hvit høyglans. Ny kjøkkeninnredning i NNNN." This results in a higher number of short and simple captions, which can be considered lazy, compared to more detailed, high-quality ones. Realtors often reuse captions because creating listings is a repetitive task, leading to similar patterns across different listings. Consequently, the dataset, which consists of realtor-written captions, is prone to overfitting toward these patterns. To prevent the model from generating only lazy captions and to avoid overfitting, we cluster the captions and select an even distribution from each

cluster for the dataset, ensuring a balanced representation from each caption class.

K-Means clustering is implemented to group captions into clusters. The parameter K, which represents the number of clusters, is chosen before the clustering process begins and is crucial for creating an optimal dataset. Selecting the correct K ensures that each cluster accurately represents a specific type of caption. K-means clustering is a widely used algorithm for text clustering. It operates iteratively, updating the centroid of each cluster with every iteration until the centroid stabilizes [56]. The result is the formation of K clusters, each representing a distinct group of similar captions. For example, sentences like "Kjøkken med plass til spisebord." and "Kjøkken med god plass til spisegruppe" should be grouped together, as should captions like "Kjøkkendetalj" and "detalje - kjøkkken." Choosing a K value that is too low will result in broad clusters that combine dissimilar captions and lose important nuances. Conversely, selecting a K value that is too high will lead to overly specific clusters, potentially breaking up similar captions into multiple groups and reducing the model's ability to generalize effectively. One can search for the optimal K value by investigating the most frequently used words for each cluster, with the goal of having the least amount of overlapping dominant words [56].

3.3 Final Datasets

For the final preprocessed datasets, the number of clusters is 5 within the categories. Looking at the variation of the captions and difference in the most frequent words within each cluster, it seemed like a suitable number. With 10 clusters, some of the clusters were much alike, and with 3 clusters, the variation of captions within each cluster was too large. There is room for improvement and a more detailed search for an optimal K-value. Then an equal amount of captions from each cluster are picked for each dataset.

The class-ratio selection consists of how many images that are desired from the three classes, "descriptive," "creative," and "informative." Not much research has been done to find an optimal ratio. However, three different datasets are selected for further use:

- **Data ID 1:** Mixed dataset, Ratio: 5.000 descriptive and 7.000 creative captions
- **Data ID 2:** Descriptive dataset, Ratio: 4000 descriptive captions
- **Data ID 3:** Creative, Ratio: 4000 creative captions

If a real estate prospect embedding is developed in further work, an informative dataset can be used, but it is excluded for now.

Method

In this chapter, we introduce the methods behind tuning, testing, and evaluation of the BLIP model. It covers some theory behind the methods of achieving a regularized model.

4.1 Hyperparameters

Methods for hyperparameter optimization are a critical component of fine-tuning a high-achieving ML model [57]. A model can require different constraints, learning rates, or regularization terms to envision different data patterns. These measures are called hyperparameters, and the parameters need to be tuned in order to improve the model accuracy and optimally solve the machine learning problem [58]. To improve the performance of a model in specific tasks, its hyperparameters, such as learning rate (LR) and batch size (BS), need to be selected [59]. The following are the selected methods that have been tested while training mainly the BLIP base model.

4.1.1 Learning Rate

Common methods for learning rate optimization include constant learning rate, step decay, and exponential decay. However, identifying the optimal learning rate for a given task usually involves a trial-and-error approach. When the learning rate is set too low, the model takes longer to converge. On the other hand, when the learning rate is too high, the model can diverge, resulting in suboptimal solutions. Finding the optimal learning rate results in the network converging in fewer iterations [59].

For this project, a search for optimal learning rate was done through trial and error. Furthermore, both LR warmup and LR decay were tested without making any improvement.

4.1.1.1 LR Warmup

Learning rate warmup is a method particularly used in deep learning. The learning rate is set to a low value in the beginning and gradually increases until it reaches the target rate. since the model struggled to converge towards good results with a high LR, a warmup method is an effective implementation to prevent an unstable start and poor convergence. A high LR at the beginning can result in huge updates which cause a destabilization of the

training process. With lower LR in the beginning, the model has a more careful exploration of weight space [10]. Using warmup, therefore, allows higher LR.

4.1.1.2 LR Decay

With LR decay implemented in the model, we get to adapt the training to have high LR for a number of iterations and then decrease over time. If the learning rate is too high, the model might overshoot and oscillate around the optimum instead of converging towards it [60].

Cosine annealing is a type of learning rate schedule that involves starting with a large learning rate, then the LR decays quickly towards a minimum value, followed by a rapid increase again. This resetting of the learning rate simulates a restart of the learning process. It is usually implemented with a *warm restart*, where the good weights are reused as the starting point.

4.1.2 Epochs

Choosing the number of epochs or iterations during model training is a simple but crucial parameter. Tuning a model involves adjusting the number of iterations in relation to the learning rate. The model should have enough iterations to converge towards the optimum value, which is usually determined by the error gradient. However, having too many iterations can cause the model to overfit the training data and learn the noise too well, resulting in poor performance on new data.

A notice is that epochs are commonly used in machine learning instead of specifying the number of iterations. This is because typically one wants to use all available data, and an epoch refers to one complete cycle through the entire training set.

$$\text{Iterations per epoch} = \frac{\text{Total number of images}}{\text{Batch size}} \quad (4.1a)$$

$$\text{Epochs} = \frac{\text{Total iterations}}{\text{Iterations per epoch}} \quad (4.1b)$$

4.1.3 Batch Size

The batch size is a crucial hyperparameter that directly affects the amount of GPU required to execute a training iteration. It represents the number of images used in one iteration. A larger batch necessitates more GPU resources, but it generally improves the efficiency and accuracy of the training process. One epoch consists of an iteration for each batch, and the number of batches is calculated accordingly [61, 62].

$$\text{Number of batches} = \frac{\text{Size of training set}}{\text{Batch size}} \quad (4.2)$$

The search for the optimal batch size in computer vision is typically constrained by the available GPU since it requires more GPUs. Also, an important consideration is that the optimal hyperparameters for a specific BS may not be the same for another BS.

4.2 Regularization

As introduced in Section 2.1, different methods, such as L2-norm regularization, can be utilized to get a well-regularized model. Furthermore, the size of the validation set can have a crucial impact on the regularization.

4.2.1 Train and Validation Split

Two different splits of the dataset are tested in this project. A 90-10 split where 90% of the complete dataset is for training and 10% for validation. And the other split is 70-30.

With a higher percentage of validation the model should in theory become more regularized, given that the training set is sufficiently large. A higher percentage of the validation set can reduce overfitting by providing a more robust assessment of the model's performance on unseen data. When more data is allocated to the validation set, the model's performance can be evaluated more accurately, revealing any tendencies to overfit the training data. This ensures that adjustments to the model's hyperparameters are based on a better representation of how it will perform in real-world scenarios, ultimately enhancing its ability to generalize to new data [63]. A rule of thumb is that when validation loss increases, it indicates that it is overfitting to the training set. Validation loss can also be employed for regularization through early stopping.

4.2.2 Early Stopping

Early stopping is a regularization method that is implemented to find an optimal solution within a given learning rate and a number of iterations. The idea behind this technique is to monitor the performance of the network on a validation set and stop the training when the validation error begins to increase or stops improving. By halting the training when the validation loss is minimized, you can prevent the model from diverging. Early stopping also reduces computational time while still allowing you to get the best model. To implement this technique, one needs to set a patience parameter, which determines the number of iterations that the model is allowed to continue without any improvement. If there is no improvement within the number of iterations specified by the patience parameter, the training process is stopped [60].

4.3 BLIP

BLIP models are used for this project. BLIP has a large and base architecture. The base model has 385 million parameters, and the large model has 470 million [64, 65]. They are both available through Hugging Face and its Transformers library. It is important to use the correct tokenizer, which is already trained for the model. Although it is pre-trained on a large set of images, it performs poorly on unseen data and, therefore, needs to be fine-tuned to classify different room categories and identify objects of interest within that room. Additionally, English is the default captioning language, whereas the goal is to generate Norwegian text, and in particular, using the style of writing of real estate agents, describing the salient features of a dwelling such that the generated text is amenable for use in a listing.

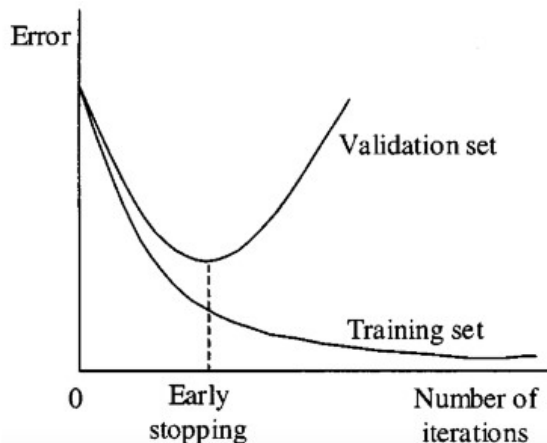


Figure 4.2.1: Example of how early stopping is used as regularization [60]

BLIP is the preferred architecture because of its relatively small GPU demand and ability to run inference on CPU [66]. BLIP can run inference using CPU, and while GPU is faster, it's not required. Furthermore, it allows for embedding which can be beneficial for further development, and it is open-source and free-licensed. Since one of the goals is to be energy efficient, the smallest option, which is BLIP base, will be the main focus.

During the training process, the model's weights are adjusted with the aim of fine-tuning them to optimal values. These updated weights differentiate the model from its default state, so it's crucial to save them for future use. When we mention different model IDs in table 4.3.1, we are primarily referring to the new weights saved with an ID.

4.3.1 BLIP License

BLIP is under the Creative Commons Attribution 4.0 International license [64]. The license states as follows. Giving the freedom to copy, redistribute, remix, transform, and build upon the material in any medium or format for any purpose, including commercial use. The licensor cannot revoke these freedoms as long as you adhere to the license terms [67].

4.3.2 Parameter Tuning Log for BLIP

The search for optimal hyperparameters often involves grid search, trial and error, or a combination of both. A grid search is a tuning technique to systematically follow a grid to find the best combination of hyperparameters. However, this is usually computationally demanding [8]. The following is a training log for BLIP, which is a partially systematic way of using trial and error that is overlooked by the developer after each session of tuning a model.

Model ID	Model size	Batch size	LR	Weight decay	Epochs	Data ID	Train-Val [%]
1	Base	4	1,0E-05	0.45	8	1	90-10
2	Base	4	1,0E-05	0.45	6	1	90-10
3	Base	7	1,0E-05	0.45	10	1	90-10
4	Base	7	1,0E-05	0.45	12	1	90-10
5	Base	7	1,0E-05	0.45	6	1	90-10
6	Base	7	1,0E-05	0.25	10	1	90-10
7	Base	7	1,0E-05	0.25	8	1	90-10
8	Base	7	5,0E-06	0.45	10	1	90-10
9	Base	7	5,0E-06	0.45	6	1	90-10
10	Base	7	1,0E-06	0.45	10	1	90-10
11	Base	7	1,0E-06	0.45	6	1	90-10
12	Base	7	1,0E-05	0.45	8	1	90-10
13	Base	7	1,0E-05	0.45	8	2	90-10
14	Base	7	1,0E-05	0.45	12	2	90-10
15	Base	7	1,0E-05	0.45	12	3	90-10
16	Base	7	1,0E-05	0.45	8	3	90-10
17	Large	3	1,0E-05	0.45	8	1	90-10
18	Large	3	1,0E-05	0.45	8	1	70-30
19	Base	7	1,0E-05	0.45	8	1	70-30
20	Base	7	2,0E-05	0.35	10	1	70-30
21	Base	7	2,0E-05	0.35	12	1	70-30
22	Base	7	2,0E-05	0.35	8	1	70-30

Table 4.3.1: A training log with the changes from the previous row highlighted in light blue. The table is divided into subtables based on the dataset, model, and batch size.

4.4 Implementation of Automatic Evaluation Metrics

As mentioned in Section 2.6, it can be challenging to evaluate the performance in text generation tasks. Since there is often no single correct answer or caption for a given image, it is difficult to measure how well a model’s output aligns with human-like text generation. The variability in what can be considered an accurate or relevant caption adds another layer of complexity, making it difficult to set fixed benchmarks or standards for assessment. A variety of methods and evaluation metrics have therefore been used to evaluate the results during the search for the optimal dataset and hyperparameters.

See Appendix C for code snippets for implementation of each automatic evaluation metric.

4.4.1 Average Sentence Length

Average sentence length is calculated with python using sentence tokenization and averaging their lengths. To calculate the average sentence length, the text is combined into one string and split into sentences. Each sentence is then split into words, and the lengths of all sentences are averaged.

4.4.2 TTR

There is no Type-Token Ratio Python package used. The implementation follows the steps from Section 2.6.1.4. It is calculated by splitting the text into words (tokens) and counting both the total number of words and the unique words (types). Then divide types over tokens, and average the scores.

4.4.3 Yule’s K

There are no public libraries used to implement Yule’s K. It is calculated using the formula in Section 2.6.1.5.

4.4.4 BLEU

The Bilingual Evaluation Understudy (BLEU) implementation of the similarity between a candidate and reference text by comparing n-grams. The reference text consists of a test set consisting of 850 realtor-written captions. The calculation uses the NLTK library in Python to calculate BLEU, including smoothing to handle variations.

4.4.5 STS

Semantic Textual Similarity *All-MiniLM-L6-v2* is the sentence transformer used to transform sentences into vectors. It maps the captions into a 384-dimensional dense vector space [54]. Furthermore, the true caption is from the same test set as for BLEU, which is 850 realtor-written captions. These vectors are compared using cosine similarity, and the average similarity score is calculated to show how closely the texts are related.

4.4.6 Overview of Metrics

- **Type-token Ration (TTR):** Measures vocabulary diversity. Higher values are better.
- **Yule’s K:** Lower values indicate a richer vocabulary.
- **Sentence Length:** Gives an idea about the syntactic complexity, but desired length is unknown.
- **Semantic Textual Similarity (STS):** Indicates how close the generated text is to some reference text. Closer to 1 is better.
- **BLEU:** A score closer to 1 is better. It is especially meant for translation tasks, so a low score may be expected in an open-ended NLG task like this.

Together, these metrics offer a comprehensive analysis of text, covering lexical diversity, structural complexity, and both lexical and semantic similarity to reference texts.

4.5 Postprocess

The BLIP model does not generate the Norwegian letter "å". Therefore, commonly used words in captions that are supposed to use "å" but have an "a" instead are replaced with the correct spelling in postprocess. Examples of this are "pa", "apen", and "ogsa" replaced with "på", "åpen", and "også". This is a very naive way of solving this problem. Therefore, *NoraLLM* was tested to postprocess sentences. However, because the inference is meant to run on CPU, the model was too large to run on CPU efficiently, and the process was therefore discarded.

Moreover, the BLIP generated captions does not include any upper-case letters. Therefore each first letter is given an upper-case letter at the beginning of the sentence.

4.6 Survey for Testing The Model

A survey was made to get a human evaluation of the tuned model. This was made out after finding the best models according to the automatic metrics used, to substantiate or disprove the evaluation results.


The target group of the product is mostly real estate agents since they write 97% of the listings [3]. It would be natural to address this survey to realtors. However, the goal is to generate captions that intrigue potential buyers who are the average citizens. This survey is therefore sent out to individuals, regardless of profession, to get as many responses as possible. The respondents are, therefore a combination of prop-tech developers, real estate agents, workers for Finn.no, students, researchers and many others.

Firstly, the survey begins with two questions to get some insight about the respondents before starting the image captioning question segments. The respondents are asked if they use AI at least once a week. This is to get an understanding of how frequently AI is used and if it can affect scores. The next question is if the respondent has seen at least one real estate listing on Finn.no or Hjem.no in the last couple of years. This is mostly to ensure that the respondents know the purpose of these captions.

4.6.1 Question Segment One

The setup is first a pairing of two questions of four images. The first question is then to decide which caption is preferred between a realtor-written and an AI-generated caption without knowing if they are human-written or not. Then the respondents are asked to guess which of the captions is AI-generated. The goal is to compare the quality of two captions and pit them against each other. Then, investigate if it is obvious to locate the AI-written caption. This measures the natural language of the generated text and its performance directly compared to the

Hvilken beskrivelse foretrekker du?




"Spisestue med åpen kjøkkenløsning"

"Kjøkkenet har god plass til spisebord ved vinduet"

Ingen av alternativene samsvarer med bildet

Begge alternativene er like gode

Hvilken av beskrivelsene er KI-generert? 1 point



"Spisestue med åpen kjøkkenløsning"

"Kjøkkenet har god plass til spisebord ved vinduet"

Vet ikke

Figure 4.6.1: Example of the two types of questions in first segment

4.6.2 Question Segment Two

The next segment is six images with multiple options of captions, and they are asked to decide which caption they prefer. The options consist of one caption written by a realtor, one by GPT-4, and one by a model from this project. This is to compare the model's performance with the benchmark AI model, GPT-4, as well as the realtor-written ones which is the current industry standard.

Hvilken beskrivelse foretrekker du?



- "Dette funksjonelle kjøkkenet har moderne hvite skap, grå benkeplater, og rustfrie hvitevarer som gir et stilrent utseende."
- "Stort og flott kjøkken."
- "Kjøkkenet er pent med moderne fronter og laminat benkeplate"

Figure 4.6.2: Example of a question in segment two

4.6.3 Question Segment Three

Lastly, the respondents are asked to rate the quality of six different captions for six images. The rating is on a scale from one to five, where one is very dissatisfied with the caption and five being very pleased with the caption. The captions that they are rating are three realtor-written and three AI-generated. This is an additional method to compare the quality of the captions. The addition of a rating scale to the existing method enhances the evaluation process by providing a quantitative measure of caption quality. This numerical approach allows for a clear comparison of how much better or worse one caption is compared to another. Simply choosing a preferred caption does not indicate how much better it is than the others. Captions might be very similar or very different in quality, and the rating scale helps to show these differences.

However, it can be less than optimal to use this type of rating because the rating from the respondents can be inconsistent, the value of three can for some be just adequate, and for others, it can be very good. The survey is therefore dependent on having many respondents to normalize the distribution of ratings.

Hvor fornøyd er du med denne beskrivelsen?

0 points



"Kjøkkenet har integrerte hvitevarer og godt med oppbevaringsplass i skuffer og skap"

1 2 3 4 5

Svært misfornøyd Veldig fornøyd

Figure 4.6.3: Example of a question in segment three

4.6.4 Data Selection

The kitchens used in the survey are randomly selected from a selection of images. A random sample is used to reduce bias in the survey questions. However, some results were removed because the realtor-written and AI-generated captions were identical. Others were excluded due to instances of AI hallucination. Additionally, captions were omitted because the realtors made spelling errors or if the captions lacked coherence when taken out of context from the real estate listing. The selected sample was picked to get a decent representation of the performance. However, human interference increases the chances of initial bias in the survey.

This is a link to the survey:

- <https://forms.gle/ggzXH1wFSeuzRAH99>

4.7 Website for User Test

One can find links to demonstration videos of the website in Appendix B

A website has been created for model inference with a user interface to gather feedback on the performance of different models. The backend API of the website contains five different models, which are BLIP models of varying sizes, with different tuning and datasets. These models are identified as 14, 15, 18, 20, and 22 from Table 4.3.1. They are the best-performing models from the BLIP base with data split of 70-30% and mixed dataset (ID 20 and 22).

CHAPTER 4. METHOD

Additionally, one model is descriptive (ID 14), one is creative (ID 15), and the last one is a BLIP Large model (ID 18). The reason for selecting these models is further explained in the results section. The application allows users to generate captions using the available models. The order of the models in the application is model ID 20, 18, 14, 15, and then 20.

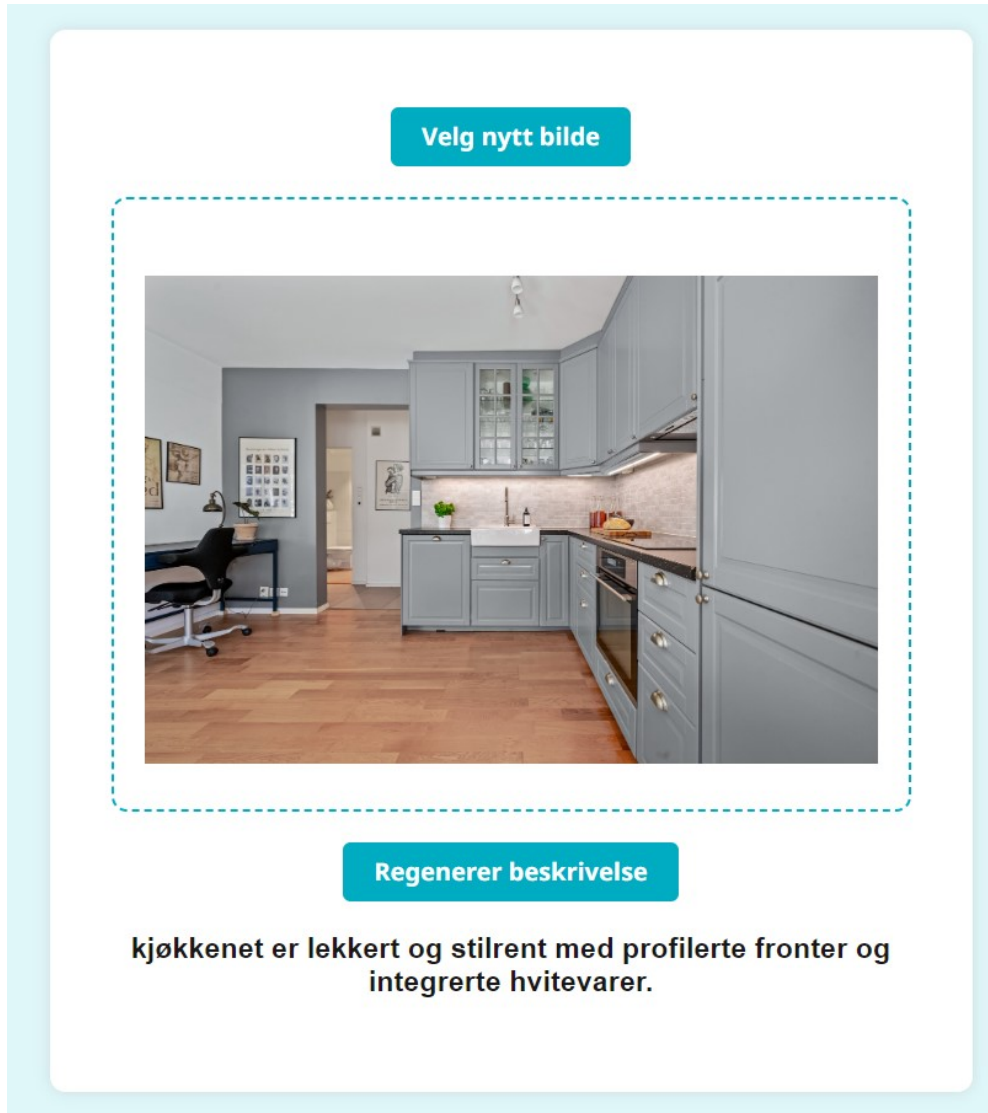


Figure 4.7.1: Screenshot from the website made for user testing

Results

The results are presented in this chapter. All of the methods introduced in Chapter 4 were tested out, and the results of the most influential methods and hyperparameters for the models' performance are the ones that are presented here. The focus is on the BLIP base model's performance on datasets consisting of kitchens.

5.1 Training BLIP

The time needed for each training session varied depending on the configuration of the hyperparameters. However, the average computation time for the different models with the mixed dataset and 70-30% data split is as follows:

Model Size	Batch Size	Average Training Time per Epoch [Min:Sec]	GPU Required [MiB]
Base	4	6:34	11177
Base	7	5:41	15835
Large	3	11:20	15587

Table 5.1.1: Training BLIP with different model and batch sizes on GPU

Using the Large instead of the Base model, the training time is almost twice as long, with approximately the same GPU usage. This illustrates how model size influences energy consumption.

5.1.1 Training and Validation Loss

This section details the results of tuning the BLIP Base model. Various hyperparameters such as learning rates, number of epochs, and L2 regularization were experimented with to achieve optimal performance.

Different Learning Rates

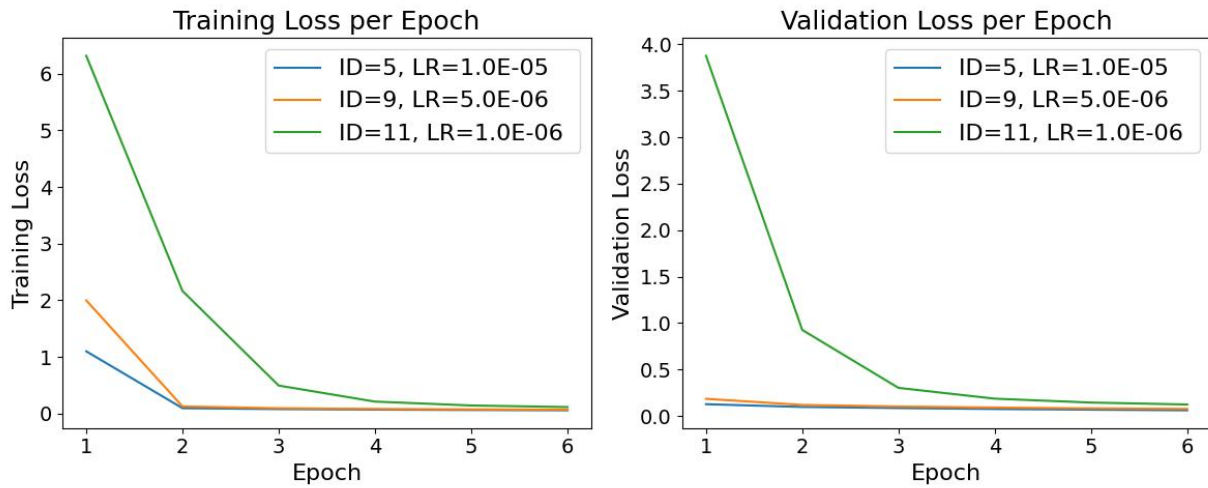


Figure 5.1.1: Training and validation loss for model ID 5, 9, and 11 where the only hyperparameter difference between them is the learning rate.

These learning rates are quite low and are typically used to ensure stable convergence, but not too low, given the rapid convergence.

Different Epochs

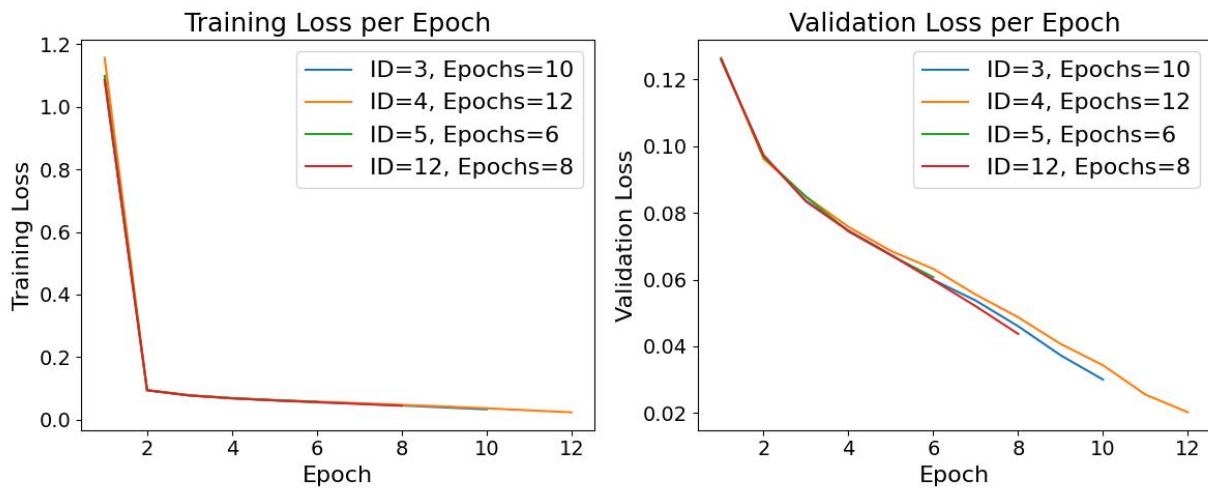


Figure 5.1.2: Training and validation loss for model ID 3, 4, 5, and 12, where the only hyperparameter difference between them is the number of epochs.

The steady decrease and stabilization in Figure 5.1.2 indicate that these epoch values are appropriate for the model. The correlation between a decrease in loss and the number of epochs proves that choosing the number of epochs influences how well one wants to fit into the training dataset.

Different L2 Regularization

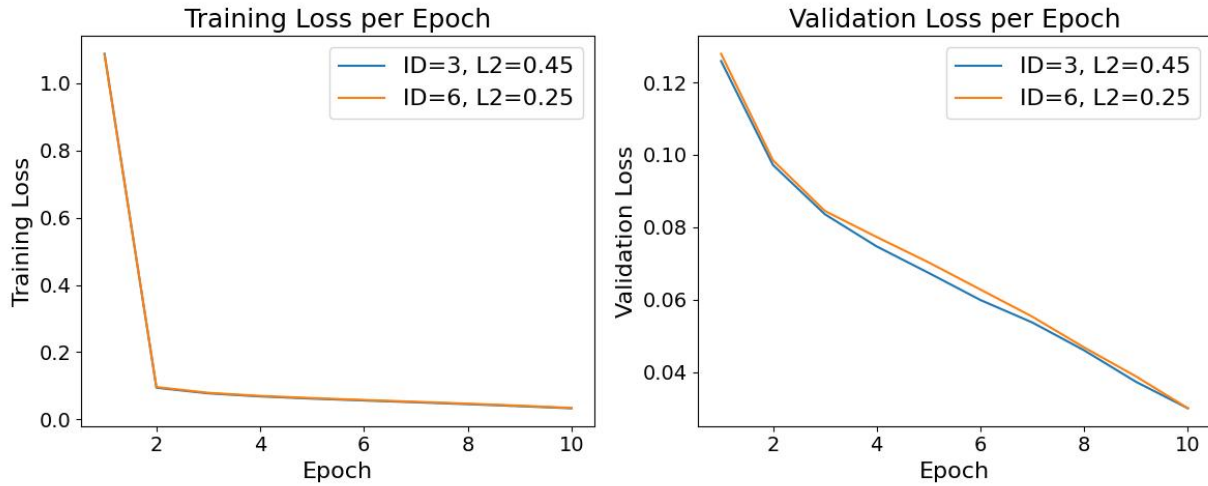


Figure 5.1.3: Training and validation loss for model ID 3 and 6 where the only hyperparameter difference between them is the L2 regularization value.

Both L2 regularization values result in very similar training and validation losses. The difference between the two is minimal, suggesting that the model is not very sensitive to these L2 values within this range. Values much higher might overly penalize large weights, while much lower values might not effectively prevent overfitting.

The learning rates, epochs, and L2 regularization values do not seem unusually high or low. The rapid convergence of training and validation losses, as well as the stability of these losses, indicate that the selected hyperparameters are within a reasonable range for the BLIP Base model on this dataset. However, the influence of different hyperparameters is not always visible in the loss calculation for training and validation, but it can still affect the further evaluation results.

5.2 BLIP Trained on Non-preprocessed Data

5.2.1 Multiple Room Types

The model trained on multiple room types effectively classifies different room types and generates suitable captions for the images. However, it performs better for some rooms, such as living rooms, bedrooms, and floor plans, compared to others. It faces challenges in providing accurate descriptions for room types with more detailed image descriptions like bathrooms and especially kitchens. The best model with non-preprocessed data was trained on 6.750 images and a validation set of 750 images, which contained floor plans, kitchens, living rooms, entrances, bathrooms, and bedrooms. Following are some examples with a model trained on this dataset. Note that the kitchen and bathroom examples in Figure 5.2.2 are amongst the best results with this model for those room types.

Bedroom



soverrommet er det god plass til dobbeltseng og godt med plass tilhørende møblement

Living Room



stuen er det god plass til sofagruppe og tv

Figure 5.2.1: Captions generated for bedroom and living room with BLIP trained on a non-preprocessed dataset for multiple room types

Bathroom



badet er det opplegg for vaskemaskin, opplegg for vaskemaskin, gulv og oppvaskmaskin

Kitchen



kjøkkenet er det godt med skap - og benkeplass til kjøkkenet er det er det er det en støblement

Figure 5.2.2: Captions generated for bathroom and kitchen

An important observation is that after selecting only the kitchen images from the set and training BLIP on that dataset, it produces almost the same captions for the same images. This gives an indication of the scalability of the model. If one is able to find good datasets for the different room types separately, then all these datasets can be merged into a model that can function for all room types.

5.2.2 Kitchen

Before preprocessing, BLIP was trained on 4200 images of Kitchens. It only wrote variations of the same four sentences.

- "kjøkkenet har en fint med skap - og benkeplass."
- "kjøkkeninnredning med profilerte fronter og benkeplass i heltre benkeplate med nedfelt oppvaskkum og kjøkkenbenk og oppvaskkum."
- "kjøkkenet er det er det god plass til spisebord"

- "kjøkkeninnredning med integrerte hvitevarer som komfyr, oppvaskmaskin og kjøøl / frys."

With this result, the concern whether the BLIP model can detect smaller objects in the image was raised.

5.3 Automatic Evaluation Results

The following results are with the preprocessed datasets, and the model IDs refer to the training log in Table 4.3.1. The models with ID 1 and 2 which are BLIP base models with batch size 4 are omitted from the results. This is because the performance is significantly worse with smaller batch size, by not being able to produce any variation of captions.

5.3.1 Mixed Class Dataset

To identify the most promising IDs for each data split, we implement a combination of the linguistic metrics, as described in Section 4.4. The dataset has the Data ID 1 in Section 3.3 containing both descriptive and creative captions. The goal is to find a combination that indicates rich and diverse vocabulary usage, as well as precisely describing the image by matching the ground truth using these automatically calculated scores.

90-10% Data Split

Model ID	TTR	Sentence Length	Yule's K	STS	BLEU
3	0,974	12,2	41,5	0,6584	0,0254
4	0,963	14,9	49,1	0,6613	0,023
5	0,973	14,6	41,1	0,6656	0,0226
6	0,969	12,2	47,1	0,6504	0,0234
7	0,934	13,7	109,4	0,6696	0,0210
8	0,971	12,5	49,2	0,6725	0,0234
9	0,969	11,9	48,8	0,6731	0,0183
10	0,981	14,8	31,7	0,6587	0,0227
11	0,827	13,5	283,0	0,6606	0,0261
12	0,972	15,2	44,2	0,6659	0,0207

Table 5.3.1: The automatic scores for BLIP Base fine-tuned on Data ID 1 with a 90-10% data split. The hyperparameters for the model IDs are found in Table 4.3.1.

The following are the most promising IDs based on the metrics scores.

ID 5: The model is closest to the median value for almost every metric. This reflects a well-balanced model tuning.

ID 10: A well-balanced model with the best vocabulary scores, and decent scores on both STS and BLEU, and good sentence length. This makes it a strong candidate for tasks needing diverse vocabulary.

Furthermore, model 11 is a classic example of an overfitted model. It is capable of achieving good scores on STS and BLEU because the test set contains many similar captions, allowing

the same caption to frequently match. However, the diversity in language is noticeably worse.

70-30% Data Split

Model ID	TTR	Sentence Length	Yule's K	STS	BLEU
19	0,985	33,7	28,0	0,6466	0,0192
20	0,979	16,3	33,5	0,6467	0,0215
21	0,970	26,3	22,4	0,6391	0,0209
22	0,986	14,9	19,4	0,6464	0,0197

Table 5.3.2: The automatic scores for BLIP Base fine-tuned on Data ID 1 with a 70-30% data split. The hyperparameters for the model IDs are found in Table 4.3.1.

ID 20: Offers a blend of syntactic complexity and high TTR, as well as having the highest BLEU score out of the models trained with the 70-30% data split. These scores describe a model that is able to generate captions with high precision and rich vocabulary.

ID 22: The model seems very promising based on the metrics. The highest TTR and lowest Yule's K scores indicate that it has high lexical diversity and rich vocabulary, as well as having a moderate average sentence length. STS and BLEU are also decent. Overall, it is a well-balanced model with good vocabulary scores.

BLIP Large

Model ID	TTR	Sentence Length	Yule's K	STS	BLEU
17	0,982	13,01	30,93	0,6636	0,0182
18	0,985	10,8	24,3	0,6509	0,0193

Table 5.3.3: The automatic scores for BLIP Large fine-tuned on Data ID 1 with both a 90-10% and 70-30% data split. The hyperparameters for the model IDs are found in Table 4.3.1.

ID 17: Good TTR and STS scores, but the lowest BLEU score out of all, and decent Yule's K. This illustrates a less accurate model with decent lexical diversity.

ID 18: Represents a balance between high vocabulary diversity in TTR and richness with low Yule's K and a high STS score, although the average sentence length is the shortest.

5.3.1.1 Data split comparison

To compare, the data split of 70-30% has better lexical diversity, and it also has longer and higher variation in average sentence length, which combined indicates a more free language and less overfitting. The STS and BLEU scores are slightly higher in the 90-10% split. The overall impression based on the automatic evaluation score is that a 70-30% split is desired for optimal performance.

5.3.2 Descriptive and Creative Datasets

Model IDs 13 and 14 are trained on the descriptive dataset, and 15 and 16 on the creative data, as described in training log Table 4.3.1.

Model ID	TTR	Sentence Length	Yule’s K	STS	BLEU
13	0,941	14,2	99,1	0,6993	0,0339
14	0,960	13,3	57,9	0,6747	0,0272
15	0,977	12,4	37,7	0,6729	0,0250
16	0,981	12,4	34,6	0,6578	0,0184

Table 5.3.4: The automatic scores for BLIP Base fine-tuned on Data IDs 2 (descriptive) and 3 (creative). The hyperparameters for the model IDs are found in Table 4.3.1.

Descriptive: Model ID 13 and 14 have the same tuning except that ID 13 is trained on 8 epochs, while ID 14 is trained on 12 epochs. ID 14 has a higher TTR and lower Yule’s K, indicating it has more vocabulary diversity and less lexical complexity. ID 13 has a higher STS and BLEU score, indicating better semantic similarity and translation quality. If vocabulary diversity and lexical simplicity are more critical, ID 14 is better. If semantic similarity and translation quality are more important, ID 13 is preferable.

Creative: ID 16 has a higher TTR and lower Yule’s K, indicating it has more vocabulary diversity and less lexical complexity. ID 15 has a higher STS and BLEU score, indicating better semantic similarity and translation quality. If vocabulary diversity and lexical simplicity are more critical, ID 16 is better. If semantic similarity and translation quality are more important, ID 15 is preferable.

Descriptive vs. Creative: The two edge cases are models 13 and 16, which are both trained on 8 epochs, but with different datasets. If you prioritize vocabulary diversity and low lexical complexity, ID 16 is the best. If you prioritize semantic similarity and translation quality, then ID 13 is the best. The two models trained on 12 epochs, which are ID 14 and 15, offer well-balanced options between these criteria.

5.4 Developer’s Analysis

There are multiple examples of generated captions in Appendix A.

During the tuning of the models, each tuning iteration is logged with a comment of how the developer experience the performance of the model. They are all evaluated with the same test dataset consisting of 850 images with captions written by real estate agents. This is the same test used to calculate the automated evaluation scores as STS and BLEU.

5.4.1 Mixed Class Dataset

Here are some highlighted overall impressions of the different models’ performances.

90-10% Data Split

Model ID	Comment
ID 5	Very overfitted and many repetitive captions. Very little coherence between the captions and images.
ID 10	Example of an overfitted model overfitted, more advanced language, but too frequently hallucinations and too many repetitions.
ID 11	The worst model. Very little variation in the language, and just generating variations of the same caption only.
ID 12	The least hallucinating model with this data split as well as producing captions that match the image and more variation in captions.

Table 5.4.1: A selection of the developer’s comments for BLIP Base fine-tuned with a 90-10% data split.

70-30% Data Split

Model ID	Comment
ID 20	A lot of variation and creativity in the captions. However, it sometimes struggles with the correct language and has frequent hallucinations.
ID 21	Good at classification, but it uses the same sentence too much - tendencies to be overfitting.
ID 22	Very little hallucinations and very well written language. Less detailed captions, but it has the most natural language.

Table 5.4.2: A selection of the developer’s comments for BLIP Base fine-tuned with a 70-30% data split.

The 70-30% was significantly better, but it was not one model that was universally best across all types of kitchen images. The best overall models were the ID 20 and ID 22. The model with ID 20 was creative but missed more often. ID 22 is a more reliable option, with simpler captions.

BLIP Large

Model ID	Comment
ID 17	This model has a very natural language, however, the image and caption do not match frequently. If this model had higher precision, then this would be a very promising model.
ID 18	This model is a more safe option with short and simple language. It still hallucinates but has much higher precision than ID 17.

Table 5.4.3: The developer’s evaluation for fine-tuned BLIP Large

CHAPTER 5. RESULTS

The only tuning difference between the models is that model 17 is trained and validated on a 90-10% data split and model 18 on 70-30%. When model 17 manages to correctly identify objects in the image, it generates well-articulated and correct captions. This happens too rarely, and is, therefore not a reliable option.

Model ID 17:



Model ID 18:



Figure 5.4.1: Examples of BLIP Large models where model 17 has a natural language and model 18 is simpler.

5.4.2 Descriptive vs. Creative

The general impression is that the majority of the time, it is difficult to distinguish between models trained on different datasets. However, both the descriptive models are better than the creative ones. Furthermore, the descriptive models are able to generate more accurate descriptions and have a higher variation of language in the captions. Out of the four models, the descriptive model with ID 14 is the best in the majority of the time. Moreover, the captions of the two creative models are very repetitive, but model ID 15 is significantly better than ID 16. To summarize, the two model options that were trained with the most epochs, models 14 and 15, were the best creative and descriptive options.

A notice for all of the four models is that they have more often poor language compared to the models trained on a larger mixed dataset.

Descriptive

Creative



Figure 5.4.2: Example of Descriptive and Creative captions generated by Model ID 14 and 15

There are more examples with comparisons of generated captions in the demonstration videos in Appendix B.

5.5 Survey Results

51 people completed the survey in the time period from 21.05.2024 to 31.05.2024. 16 out of the 51 respondents do not use AI at least once a week, and the remaining 35 use AI at least once a week. Furthermore, 100% said that they had seen at least one real estate advertisement at Finn.no or Hjem.no in the last two years.

5.5.1 Question Segment One

The first segment of the survey results indicates the participants' preferences and ability to detect AI-generated captions. A high average preferring AI-written captions and a low average of correctly guessing which caption is AI-written would indicate that the model is able to demonstrate human intelligence and generate captions with natural language.

Which type of caption is preferred (Average Score)

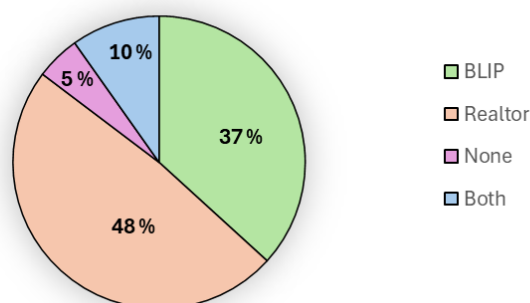


Figure 5.5.1: A pie chart illustrating the preferred caption type between BLIP and Realtor out of 51 respondents

The pie chart in Figure 5.5.1 reveals that 48% of respondents preferred captions generated by realtors, 37% preferred BLIP-generated captions, 10% liked both, and 5% had no preference. That the realtor-written captions are preferred over AI-written ones less than half of the times means that the BLIP model is able to generate matching captions.

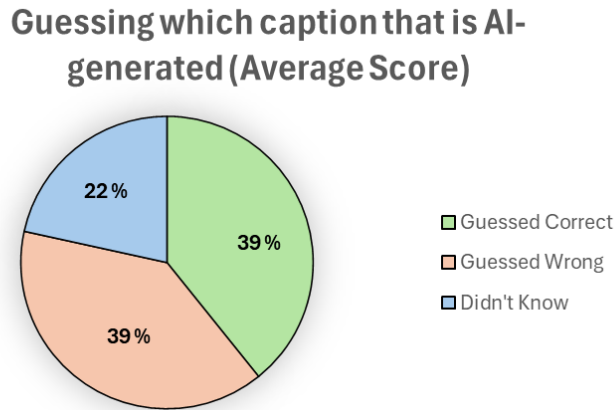


Figure 5.5.2: A pie chart showing the score from 51 respondents guessing which caption that is AI generated

In terms of identifying which captions were AI-generated, the results in Figure 5.5.2 show that participants were equally split between correctly identifying AI-generated captions (39%) and incorrectly identifying them (39%), with 22% of participants unsure. This low average of correct identification suggests that the AI-generated captions are effective in mimicking human-written language, making them challenging to distinguish.

Furthermore, the chi-squared test is used to investigate the correlation between the respondents' use of AI and how well they perform at identifying AI-written captions using the results in Table 5.5.1. The chi-squared test is a statistical method used to determine if there is a significant association between two categorical variables. It compares the observed frequencies in a table to the frequencies we would expect if the variables were independent. If the observed and expected frequencies are significantly different, it suggests an association between the variables [68]. The hypothesis is that if one uses AI regularly, then the person is expected to more often identify the AI-generated captions.

Score	Uses AI	Does Not Use AI
0	5	3
1	10	7
2	11	4
3	9	2
4	0	0

Table 5.5.1: Identifying AI-generated captions survey scores. The score is out of 4 images.

The score in Table 5.5.1 refers to how many times the respondent correctly identified an AI-generated caption. Using these scores, the chi-square test statistic is approximately 1.942, and the p-value is approximately 0.585. Since the p-value is much greater than the common significance level of 0.05, we fail to reject the null hypothesis. This means there is still no

statistically significant correlation between the use of AI and the scores.

5.5.2 Question Segment Two

In the second segment, preferences among different types of captions were measured again, but now including captions generated by GPT-4.

**Which type of caption is preferred
(Average Score)**

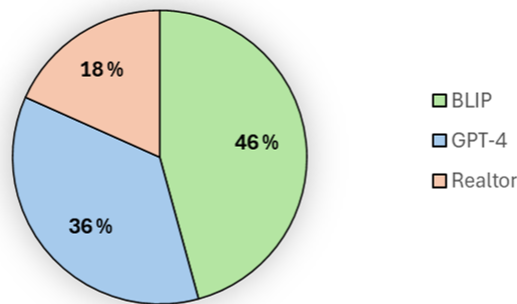


Figure 5.5.3: A pie chart illustrating the preferred caption type between BLIP, GPT-4 and Realtor out of 51 respondents

Figure 5.5.3 demonstrates that 46% of respondents preferred BLIP captions, 36% preferred GPT-4 captions, and 18% favored realtor captions. This shift in preference towards AI-generated captions over human-generated ones indicates a high level of acceptability for AI-generated content among participants.

5.5.3 Question Segment Three

The third segment provides a comparison of scores from the scale rating from 1 to 5 for each type of caption. The AI-written ones are from the BLIP model with ID 20.

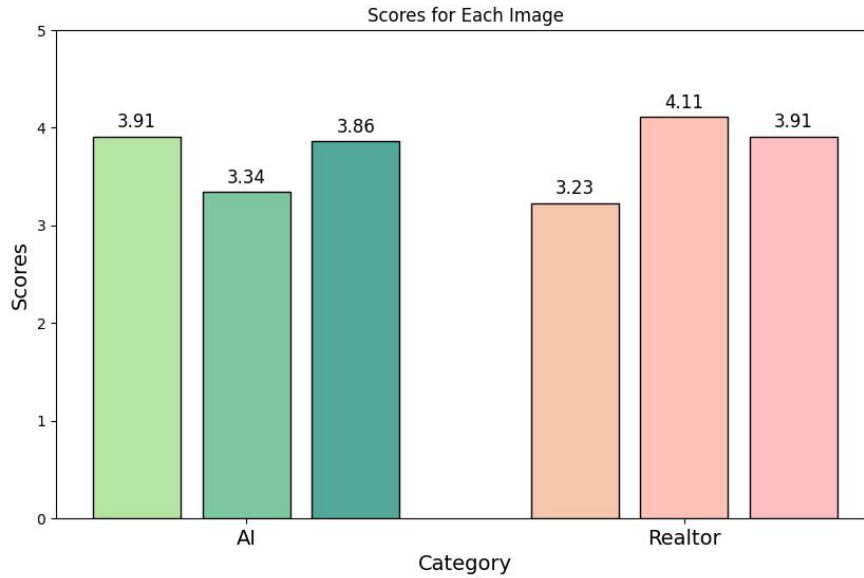


Figure 5.5.4: A bar chart showing the average caption-quality score for each caption

Figure 5.5.4 illustrates that AI-generated captions are rated highly, nearly on par with realtor-generated captions.

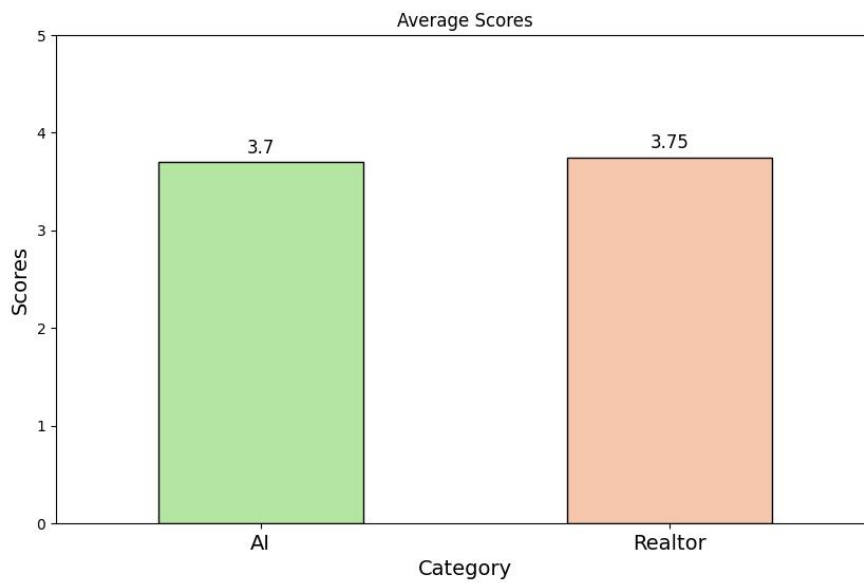


Figure 5.5.5: A bar chart showing the total average of the average caption-quality

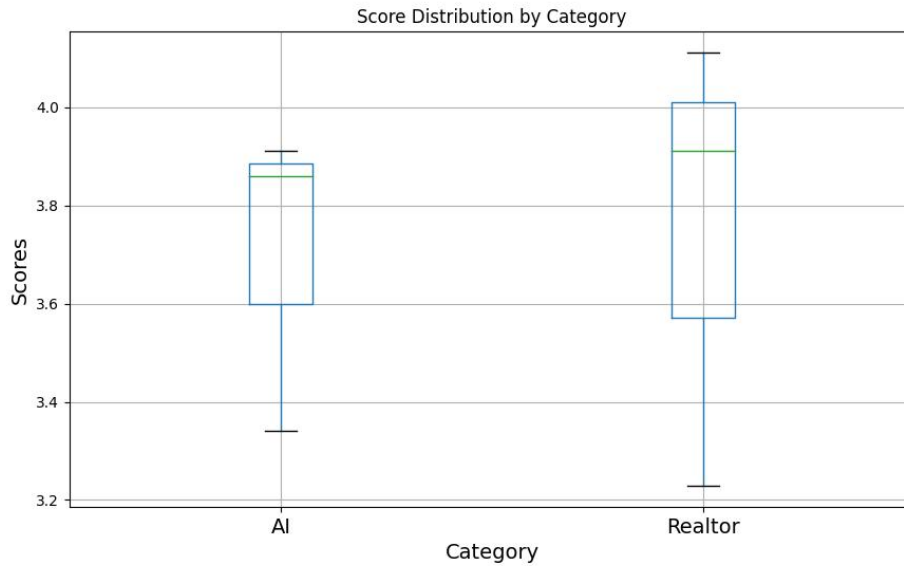


Figure 5.5.6: A box plot for average caption-quality scores showing median and variance

Further analysis in Figure 5.5.5 shows the average scores for AI-generated captions (3.70) and realtor-generated captions (3.75), suggesting minimal difference in perceived quality between the two. Figure 5.5.6, a box plot of score distribution, reveals that the distribution of scores for AI-generated captions is narrower and slightly lower compared to realtor-generated captions, which have a wider distribution and a slightly higher median score.

5.6 User Test Results

One can find links to demonstration videos of the website used for user test in Appendix B

The test was conducted 03.06.2024 from 15:55 to 17:15. The user is a woman in the age of 26. She owns her own apartment and is familiar with both the buying and selling process of an apartment, which makes her a suited representative for the test. The following is her feedback on the website.

"Charming website, good color choice, and intuitive navigation. The text generated by all the models is generally well-formulated and structured. Minor nitpicks here and there, with some models having incorrect "detections" of what is in the image, and some have minor issues with typos and sentence structure. All in all, it is a user-friendly website that generates good suggestions for image captions for a real estate ad. Overall, I really liked the fifth model (ID 22) due to its lively language. The third model (ID 14) was the second best since it managed to describe small details in the image that I did not detect before it was pointed out by the model."

Model ID	Comment
ID 20	Mostly very accurate, but occasionally poor sentence structure and sometimes describing things that are not in the picture.
ID 18	A bit repetitive, and some wrong observations.
ID 14	Describing the image well. Very accurate and observant, even on unclear images. It notices smaller details.
ID 15	The worst. It is very repetitive, and there are several occasions of poor sentence structure.
ID 22	Prefer this model most of the time. It had cozy descriptions and a positive sentence structure. I would imagine this being the best for selling purposes

Table 5.6.1: The user test’s comments for the five different models.

5.7 GPT-4

GPT-4V is the benchmark model for the majority of image captioning tasks in most languages. The following results are with a prompt asking for a caption written as a realtor, keep it simple, and only use one sentence generated with GPT-4. This prompting for generative AI is often called role prompting.

Thus, with good results, it is not possible to do the same tests for GPT-4 as for BLIP within a reasonable time because there is a usage cap on GPT-4. It dynamically adjusts depending on demand and system performance in practice, but it usually sets a cap to 30-40 messages every 3 hours [69].

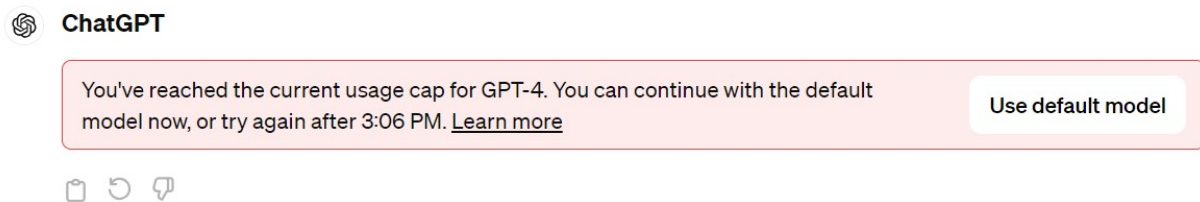


Figure 5.7.1: Screenshot of usage cap error-message in GPT-4

5.7.1 Inference Speed Comparison

An important part of the results of generative AI is the inference time of each model. On average, GPT-4 Turbo had an inference time of approximately 16 seconds, hosted on a T100 GPU server. Furthermore, OpenAI launched an updated version on the 13th of May 2024, called GPT-4o, which is supposed to generate tokens two times faster than GPT-4’s previous update [70]. It had an average inference time of approximately 7.5 seconds. Both these tests were done on 20 images of kitchens. To compare, the BLIP base model can generate more than 400 captions running on GPU at the same time. The BLIP base model is significantly faster.

Furthermore, BLIP can be hosted on CPU. The inference time with BLIP running on the CPU introduced in Section 1.3.1 is approximately four seconds. It is sometimes shorter and sometimes longer, depending on the model size and number of tokens detected for the specific image.

5.7.2 Caption Comparison GPT-4 and BLIP

Here are some examples of the performance of GPT-4 compared to BLIP. Note that these were also used in the survey.



Figure 5.7.2: Example of captions generated by GPT-4 and BLIP model with log ID 20

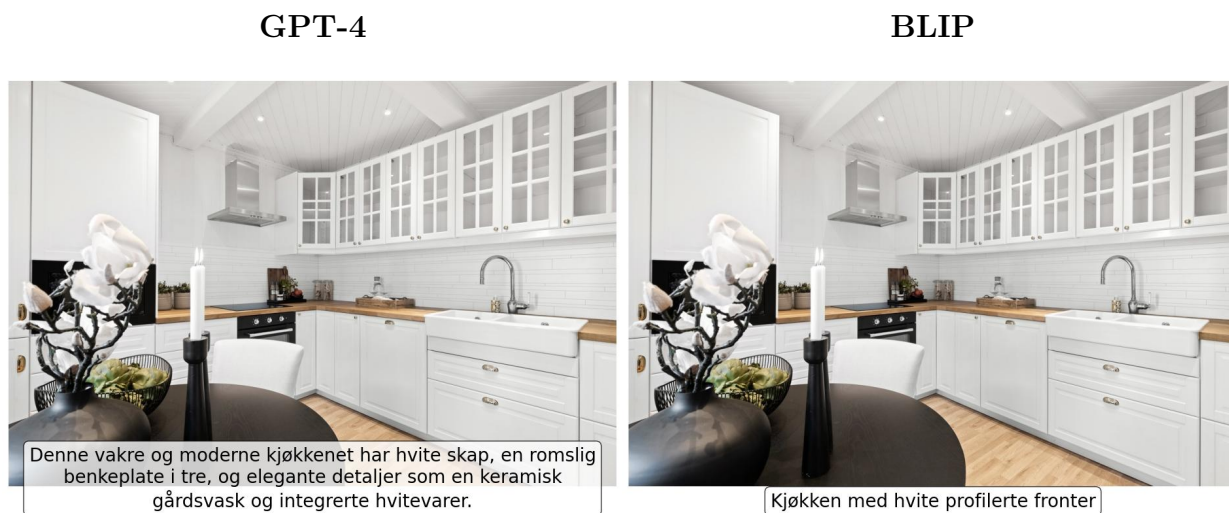


Figure 5.7.3: Example of captions generated by GPT-4 and BLIP model with log ID 20

As one can see in both Figure 5.7.2 and Figure 5.7.3, both of the models are able to write suited captions. Despite the fact that the prompt for GPT-4 said to keep it short and simple, it tended to generate longer sentences that were more descriptive.

Chapter 6

Discussion

In this chapter, the results are discussed, and limitations, improvements and options for further work are introduced.

6.1 Dataset

6.1.1 Comment on Preprocessing

Before implementing any preprocessing steps other than data cleaning, it was discovered that it was possible to fine-tune BLIP to generate captions and have Norwegian as the default language. Additionally, it was demonstrated that the model could classify different room types and generate captions that mimic the language of realtor-written captions. However, BLIP struggled to capture important details and objects in the image, making the project seem less feasible with the available data. Therefore, to further preprocessing the dataset proved to be the most vital part of this project. This chosen way of constructing the dataset was essential for achieving a high-performing image captioning model. Although the dataset and tuning methods combined resulted in better captions, there is still room for improvement.

Firstly, the work done on cleaning the text, such as removing spelling errors and names, can be further improved. There are examples of kitchen and appliance suppliers that are being named in the generated caption. E.g. "Integrerte hvitevarer fra Siemens", where Siemens should have been "XXX", this is a result of a poorly cleaned dataset.

Moreover, the dataset can be selected in many different ways. The K-value defining the number of clusters impacts how balanced the dataset is. Choosing the optimal K can prevent an early overfit and contribute to a diverse language. Moreover, the class ratio between descriptive, creative, and informative data can decide a lot about the sentence structure and the wording of the captions. For further work, one should investigate the dataset parameters. Both finding the optimal number of clusters within each class as well as finding the optimal class ratio between descriptive, informative, and creative in the mixed dataset.

6.1.2 Descriptive vs. Creative

The creative model is not as creative as intended, but the descriptive is performing well. There is a natural reason for this. It is easier to fit a model to captions that actually describe the objects, and therefore, the descriptive model was expected to perform better. An example is during training when it is a picture of a green cabinet and wooden desk, and the model generates a caption describing this. For the descriptive dataset, the captions are more likely to have a ground truth saying it is a green cabinet and a wooden desk, which makes it easier for the model to fit this caption. On the other hand, if the model had a creative ground truth, saying it is a warm and cozy kitchen, then it would be more difficult to tune the model with this. Unless the true captions consistently described those types of kitchens as warm and cozy.

Although the descriptive being the better model, the descriptive and creative data did not always produce significantly different captions. This is commented on in Section 5.4.2, and several factors can be the reason why the difference between descriptive and creative is not even larger. Firstly, the sentence transformer in Section 3.2.3 for caption classification is trained on small manually labeled data that might contain inconsistencies and errors in labeling. Secondly, the classes are not agnostic, which is a challenge when classifying the captions. This means that the model has difficulties classifying the captions since there is not a clear separation between the different classes. Lastly, the models are tuned on a smaller and less diverse dataset compared to the best-performing models. The size of the dataset proved to affect both the precision and language quality of the generated captions. Further improvement is, therefore, to collect more data and enlarge both of the datasets. This would improve the language and could also increase the differences between descriptive and creative models.

6.2 Hyperparameters Selection

There were still tendencies of overfitting, even with the added preprocessing steps. It was, therefore, important to search for optimal hyperparameters.

Epochs: The best way to observe the difference with different epochs was with human evaluation. The validation loss decreased when the epochs increased, which is a healthy sign during training. However, by looking at the test results, the best models came from training on fewer epochs, since the model started to overfit with higher numbers of epochs. Furthermore, early stopping was implemented, but unnecessary because the model started to overfit before the selected patient stopped the training process. Additionally, the optimal number of epochs was dependent on which dataset that was used. For the creative and descriptive datasets, the best models were trained on 12 epochs, instead of 8.

Learning Rate: It needed to be low throughout training and proved to be an important hyperparameter to select. Too high LR made the model very little detail-oriented, and too small LR forced the model to overfit. Any LR schedules as warmup or decay only hindered getting a generalized model.

Data Splitting: It was crucial for creating a well-regularized model. A 30% validation set led to a more diverse vocabulary and reduced fixation on a few sentences, thus better at avoiding repeated variations of the same sentences.

Batch Size: Increasing the batch size resulted in much better generated captions, both in terms of language diversity and classification precision. The thesis therefore only focus on the largest batch sizes possible with the available GPU.

Model Size: Having different model sizes made a difference since one of the best performing models was with the large BLIP model, thus using significantly less time on tuning that model. Also, the inference time is noticeable longer with a larger model.

6.3 Automatic Evaluation

The most prominent way to automatically evaluate the model’s performance turned out to be a balanced approach across the different metrics. The edge cases resulted in either hallucinations and non-accurate descriptions or poor language. A combination of all of the metrics to evaluate the model gave insight and understanding of model performance. Although, it did not provide a clear understanding of which of the tunings that did the best. Therefore, these metrics should be combined with the manual testing of the models.

A challenge with automatic evaluation methods such as STS and BLEU is that the true captions used in this task are sentences written by realtors. This means that the quality of the ground truth can vary depending on the realtors’ efforts. The real goal is not necessarily to write the same as realtors. Therefore, receiving a low score does not always mean that the generated caption is incorrect. It simply means that it is not similar to the realtor-written one, but this is not captured by automatic evaluation. Furthermore, the BLEU score is very low for every fine-tuning. This reflects the issue with using n-gram co-occurrence on an open-ended text generation task.

6.4 Human Evaluation

A developer’s interaction between each training iteration was beneficial because it allows for immediate adjustments and fine-tuning of the model parameters based on observed performance. This enhanced overall model accuracy and efficiency. Additionally, this iterative feedback loop helps identify and rectify any emerging issues or biases early in the training process, leading to achieving a reliable model faster. A Survey is a good way to get many respondents with comparable answers, while user tests are beneficial for receiving high-quality and detailed feedback.

6.4.1 The Developer’s Evaluation

Firstly, there is a high correlation between which models were considered best and worst based on the automated metrics scores and the human evaluation. However, it does not capture the small nuances in language that differ between a satisfactory caption and not. For example, an important notice is that the balanced options turned out to be the preferred options, which we learned from human evaluation. E.g. in Section 5.3.2 where Model 14 and 15 were the two balanced options out of the four possible descriptive and creative options, and they were also the preferred models based on the developer’s evaluation. Moreover, models with balanced metric scores could also vary in quality. Both ID 14 and 15 were very

similar in having balanced out the metric scores. However, the descriptive model with ID 14 proved to be significantly better. This emphasizes the importance of using all selected evaluation metrics for this project. Additionally, human oversight is necessary to fine-tune the results. With the developer's interaction, one can detect anomalies and wrongly written captions for each iteration of training.

6.4.2 Survey Results Discussion

The survey was a good addition to the evaluation process to confirm the evaluation done with automatic metrics and by the developer, and to ensure that the developer bias is not too large. Overall, it gave a good insight into general performance. It is challenging to get a good representation of the model with only 20 questions, but it was adequate to get a general impression of the performance.

In segment one, less than half of the time, the respondents were able to correctly guess if it was written by an AI. This had only two caption options, one of which was always written by an AI and one by a realtor. The fact that the respondents, the majority of the time, were not able to recognize AI-generated captions indicates that the model generates captions with a natural language that sounds human. Furthermore, in this segment, the realtor-written being preferred less than half of the time indicates that the model is a suited substitute and a usable tool for realtors.

In segment two, the realtor-written captions are only preferred 18% of the time on average. It reinforces the impression that AI-generated captions are accepted for this task. Furthermore, with the BLIP model being preferred 46% and GPT-4 36% of the time, the fine-tuned BLIP model is competitive against the benchmark model, GPT-4. This means that BLIP is superior in terms of having a lower carbon footprint, but it is also faster and often generates better captions.

In segment three, the variance in ratings in Figure 5.5.6 is higher for captions written by realtors compared to those generated by BLIP. This could be due to the model being trained on a balanced dataset, resulting in more consistent captions with fewer extremes in quality. The product aims to provide accurate and adequate captions as a tool and baseline for realtors, potentially reducing the occurrence of *lazy* captions. However, the most detailed and high-quality captions are still more likely to be written by realtors themselves.

The impression of the performance after seeing the results is that using AI is suited to replace the repetitive task of captioning images for real estate listings. Furthermore, a fine-tuned BLIP mode is able to compete with the benchmark, which is the general GPT-4 model.

6.4.3 User Test

Although a survey is a suitable way to get a higher quantity of feedback and a general impression of the performance, an application for user test resulted in a more thorough evaluation.

An interesting observation made with the user test is that the best-performing BLIP Large model with ID 18 was considered as one of the most promising models before user testing. After a thorough and more extensive test of the model, it proved to be less impressive by

frequently repeating itself and hallucinating.

Additionally, the models with IDs 20 and 22 are two very similar BLIP base models. The only difference between them is the number of epochs. They were considered to have very similar performance prior to the user test, however, the user experienced model ID 22 to be significantly better. The argumentation is, amongst others, based on the feeling of a natural and warm language that is suited for this product's purpose. This type of argumentation from the user adds a new dimension to the evaluation system.

The user was also able to detect the descriptive model. This illustrates that there is a recognizable difference between the creative and descriptive classes.

6.5 Comment on Best Performing Models

The selected IDs 14 (Descriptive), 18 (Large), 20, and 22 each bring distinct strengths to the table, from rich vocabulary to syntactic complexity and diversity. This makes them suitable for different types of linguistic analysis or generation tasks depending on specific needs such as richness, complexity, or diversity. However, a descriptive model with ID 14 is the most detail-oriented, despite being trained on a smaller dataset. Furthermore, the BLIP Base model with tuning ID 22, which is trained on a larger mixed dataset on only 8 epochs, produced the most generalized results. It was not found a model that was universally best across all types of images, so it is too early to conclude with one superior model. On the positive side, it proves that both tuning hyperparameters and using different datasets can constitute a significant difference in the results.

6.6 Limitations

6.6.1 GPU Limitations

The training environment was limited due to GPU constraints. The servers are costly, and this project utilized a GPU server owned by a company that also had other uses for it. A combination of late access to the server, and the need for Solgt.no to use the server as well meant that the fine-tuning was completed within a limited time window. Without this limitation, or with a server equipped with more GPUs, it would be possible to further tune the BLIP Large model and other larger models. Additionally, hyperparameter tuning could be performed on them, providing a better foundation for comparison.

6.6.2 User Test Limitations

A limitation of the user test is that there was only one test person. It is preferable to have more test users and, ideally, have at least one real estate agent amongst them. This limitation is caused by the fact that the website is not publicly available. Deploying this website facilitates more user tests, however, the cost of hosting a website to run these models through Amazon Web Services is too high. This limits the test to only have participants who are physically present at NTNU, where the computer hosting the website is located. Therefore, only one person tested the model other than the developer.

6.7 Future Work

6.7.1 Dataset

The preprocessing has laid the foundation for searching for the optimal dataset. Both to increase the data size and then have it go through the preprocessing pipeline in an efficient way. This means finding the optimal data size, choosing optimal dataset parameters for the number of clusters within each class, and searching for the optimal class-ratio between descriptive, informative, and creative.

Moreover, a dataset with all room classes should be made. This necessitates a labeled dataset for language cleaning and another labeled dataset for classified captions within each class. The working proof of concept with kitchens indicates that thorough work on the multi-class dataset will result in a high-performing image captioning model for every image in a real estate listing.

6.7.2 Embedding

Implementing an embedded system that combines textual information from prospects with an image captioning model can result in the generation of more informative captions. By tokenizing the text in the same manner as the objects in the image, more detailed information about the listing can be provided. An improvement would be to include both the year and kitchen brand in the automated caption, instead of generating "XXX" and "NNNN." Both the name of the kitchen supplier and the renovation year can often be found in the real estate prospect.

6.7.3 Extract Structured Data

One can further utilize a model being trained to describe images from real estate listings. The images provide useful information about the dwelling that is not written in the listing. Examples of this can be classifying an apartment with an open floor plan solution with a kitchen island and space for a dining table. One can then do feature research to find if the objects in that description can increase or decrease a dwelling's value. This is another reason why a descriptive dataset is important so that the generated text is as detailed and accurate as possible.

Conclusion

This thesis concludes that using transformers to automatically generate Norwegian image captions for real estate listings is a feasible task. The BLIP model has proven to be beneficial for adapting to Norwegian, being energy efficient, and generalizing well on the given data.

A crucial part of the project was constructing a sufficient dataset, as the original set had spelling errors and irrelevant captions and was highly unbalanced. By cleaning, classifying, and clustering captions, we obtained a dataset that was suitable for tuning a generalized model. The key findings indicate that the BLIP model, when fine-tuned with this refined dataset, can produce captions that are both accurate and contextually relevant to the real estate domain. Classifying the data into different types of captions, such as descriptive and creative proved to enhance the precision and capturing important nuances of the language for the generated captions.

Moreover, comparing the evaluation results proves that the selected automatic evaluation metrics are able to provide useful insight into the models' performance but should be employed in combination with human evaluation methods. Furthermore, the survey and user test combined illustrated that a fine-tuned BLIP model is able to generate captions that were preferred over realtor-written ones, proving a natural language and precise object detection.

In conclusion, this project successfully develops a model that automates manual tasks in the real estate industry, improving efficiency and supporting the adoption of AI-driven solutions in Norwegian real estate listings. Future work will focus on enhancing the model's performance by doing further research on the dataset and train the model on a data consisting of every room type in a listing. Additionally, improve the application with extended embedding and extracting structured data.

References

- [1] *Solgt.no: Få et bud på din bolig innen 48 timer.* en. URL: <https://solgt.no/> (visited on 12/06/2023).
- [2] *Largest increase in foreign direct investment in real estate and manufacturing.* en. Jan. 2021. URL: <https://www.ssb.no/en/utenriksokonomi/artikler-og-publikasjoner/largest-increase-in-foreign-direct-investment-in-real-estate-and-manufacturing> (visited on 11/27/2023).
- [3] *Visste du dette om FINN eiendom?* nb. URL: <https://www.finn.no/realestate/artikler/finns-boligtips/visste-du-dette-om-finn-eiendom> (visited on 05/30/2024).
- [4] Cindy Gordon. *ChatGPT And Generative AI Innovations Are Creating Sustainability Havoc.* en. Section: AI. URL: <https://www.forbes.com/sites/cindygordon/2024/03/12/chatgpt-and-generative-ai-innovations-are-creating-sustainability-havoc/> (visited on 04/14/2024).
- [5] *Goal 9 | Department of Economic and Social Affairs.* URL: <https://sdgs.un.org/goals/goal9> (visited on 12/18/2023).
- [6] *Satsing på kunstig intelligens.* nn. URL: <https://www.forskningsradet.no/forskningspolitikk-strategi/ltp/kunstig-intelligens/> (visited on 06/03/2024).
- [7] *Kunstig intelligens i Norge - nytte, muligheter og barrierer.* no. Jan. 2024. URL: https://www.nho.no/tema/digitalisering/Kunstig_intelligens_i_Norge_Rapportsammendrag_S0A2023/ (visited on 04/14/2024).
- [8] *What is Grid Search? | Grid Search Defined | Dremio.* URL: <https://www.dremio.com/wiki/grid-search/> (visited on 06/17/2024).
- [9] *Neural networks: activation functions.* en. July 2017. URL: <https://www.jeremyjordan.me/neural-networks-activation-functions/> (visited on 06/05/2024).
- [10] Jon Krohn. *Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence.* Addison Wesley Data & Analytics Series. Pearson Addison-Wesley, 2020.
- [11] Jason Brownlee. *Gentle Introduction to Vector Norms in Machine Learning.* en-US. Feb. 2018. URL: <https://machinelearningmastery.com/vector-norms-machine-learning/> (visited on 12/23/2023).
- [12] *Transfer learning and fine-tuning | TensorFlow Core.* en. URL: https://www.tensorflow.org/tutorials/images/transfer_learning (visited on 06/05/2024).
- [13] *Image captioning.* URL: https://huggingface.co/docs/transformers/main/en/tasks/image_captioning (visited on 05/04/2024).

REFERENCES

- [14] Nico Klingler. *CLIP: Contrastive Language-Image Pre-Training (2024)*. en-US. Dec. 2023. URL: <https://viso.ai/deep-learning/clip-machine-learning/> (visited on 05/01/2024).
- [15] Akriti Upadhyay. *Unveiling the Power of Multimodal Language Models in Image Captioning*. en. Mar. 2024. URL: <https://medium.com/@akriti.upadhyay/unveiling-the-power-of-multimodal-language-models-in-image-captioning-970932de0e5f> (visited on 04/24/2024).
- [16] Rishi Bommasani et al. *On the Opportunities and Risks of Foundation Models*. en. arXiv:2108.07258 [cs]. July 2022. URL: <http://arxiv.org/abs/2108.07258> (visited on 04/26/2024).
- [17] Ashish Vaswani et al. *Attention Is All You Need*. arXiv:1706.03762 [cs]. Aug. 2023. URL: <http://arxiv.org/abs/1706.03762> (visited on 04/24/2024).
- [18] Navaneeth Malingan. *What is Decoder in Transformers*. en. Apr. 2023. URL: <https://www.scaler.com/topics/nlp/transformer-decoder/> (visited on 05/02/2024).
- [19] Rokon. *RNN vs. LSTM vs. Transformers: Unraveling the Secrets of Sequential Data Processing*. en. Sept. 2023. URL: <https://medium.com/@mroko001/rnn-vs-lstm-vs-transformers-unraveling-the-secrets-of-sequential-data-processing-c4541c4b09f> (visited on 04/24/2024).
- [20] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv:2010.11929 [cs] version: 2. June 2021. URL: <http://arxiv.org/abs/2010.11929> (visited on 04/24/2024).
- [21] Md Zakir Hossain et al. *A Comprehensive Survey of Deep Learning for Image Captioning*. arXiv:1810.04020 [cs, stat]. Oct. 2018. DOI: 10.48550/arXiv.1810.04020. URL: <http://arxiv.org/abs/1810.04020> (visited on 06/17/2024).
- [22] *Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network*. en-US. URL: <https://www.upgrad.com/blog/basic-cnn-architecture/> (visited on 12/05/2023).
- [23] Sai Balaji. *Binary Image classifier CNN using TensorFlow*. en. Aug. 2023. URL: <https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697> (visited on 12/12/2023).
- [24] *Using transformers at Hugging Face*. URL: <https://huggingface.co/docs/hub/transformers> (visited on 04/25/2024).
- [25] *LMSys Chatbot Arena Leaderboard - a Hugging Face Space by lmsys*. URL: <https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard> (visited on 06/03/2024).
- [26] Mohammed Lubbad. *The Ultimate Guide to GPT-4 Parameters: Everything You Need to Know about NLP's Game-Changer*. en. Oct. 2023. URL: <https://medium.com/@mlubbad/the-ultimate-guide-to-gpt-4-parameters-everything-you-need-to-know-about-nlps-game-changer-109b8767855a> (visited on 06/10/2024).
- [27] *norallm/normistral-7b-warm · Hugging Face*. Feb. 2024. URL: <https://huggingface.co/norallm/normistral-7b-warm> (visited on 06/08/2024).
- [28] *Sustainable future - LUMI*. URL: <https://www.lumi-supercomputer.eu/sustainable-future/> (visited on 06/08/2024).

REFERENCES

- [29] *Introducing Stable Video Diffusion*. en-GB. URL: <https://stability.ai/news/stable-video-diffusion-open-ai-video-model> (visited on 05/23/2024).
- [30] Junnan Li et al. *BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation*. en. arXiv:2201.12086 [cs]. Feb. 2022. URL: <http://arxiv.org/abs/2201.12086> (visited on 03/12/2024).
- [31] Nadeem. *Cost Function & Loss Function*. en. Dec. 2021. URL: <https://nadeemm.medium.com/cost-function-loss-function-c3cab1ddffa4> (visited on 12/23/2023).
- [32] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. *Evaluation of Text Generation: A Survey*. en. arXiv:2006.14799 [cs]. May 2021. URL: <http://arxiv.org/abs/2006.14799> (visited on 05/19/2024).
- [33] *BLEU*. en. Page Version ID: 1221925142. May 2024. URL: <https://en.wikipedia.org/w/index.php?title=BLEU&oldid=1221925142> (visited on 05/31/2024).
- [34] *IBM Research*. en. URL: <https://dominoweb.draco.res.ibm.com/dominoweb.draco.res.ibm.com/5c651a88cb24938185256acb0055e548.html> (visited on 05/31/2024).
- [35] Chin-Yew Lin and Eduard Hovy. “Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics”. In: *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. 2003, pp. 150–157. URL: <https://aclanthology.org/N03-1020> (visited on 05/26/2024).
- [36] *Understanding BLEU Score in NLP: Evaluating Translation Quality*. en-GB. URL: <https://codelabsacademy.com/blog/understanding-bleu-score-in-nlp-evaluating-translation-quality> (visited on 05/31/2024).
- [37] *Semantic textual similarity (STS) and search*. en. May 2023. URL: <https://algolia.com/blog/product/semantic-textual-similarity-a-game-changer-for-search-results-and-recommendations/> (visited on 05/06/2024).
- [38] *The quick brown fox jumps over the lazy dog*. en. Page Version ID: 1225665462. May 2024. URL: https://en.wikipedia.org/w/index.php?title=The_quick_brown_fox_jumps_over_the_lazy_dog&oldid=1225665462 (visited on 05/26/2024).
- [39] Dax Thomas. “Type-token Ratios in One Teacher’s Classroom Talk: An Investigation of Lexical Complexity”. en. In: ().
- [40] *Lexical diversity*. en. Page Version ID: 1139362908. Feb. 2023. URL: https://en.wikipedia.org/w/index.php?title=Lexical_diversity&oldid=1139362908 (visited on 04/30/2024).
- [41] Kumiko Tanaka-Ishii and Shunsuke Aihara. “Computational Constancy Measures of Texts—Yule’s K and Rényi’s Entropy”. In: *Computational Linguistics* 41.3 (Sept. 2015), pp. 481–502. ISSN: 0891-2017. DOI: 10.1162/COLI_a_00228. URL: https://doi.org/10.1162/COLI_a_00228 (visited on 06/19/2024).
- [42] *The Turing Test: What Is It, What Can Pass It, and Limitations*. en. URL: <https://www.investopedia.com/terms/t/turing-test.asp> (visited on 05/24/2024).
- [43] *Uncanny Valley - an overview | ScienceDirect Topics*. URL: <https://www.sciencedirect.com/topics/computer-science/uncanny-valley> (visited on 06/10/2024).
- [44] Laura Carnevali. *Understanding Hallucinations in AI: A Comprehensive Guide | Pinecone*. en. URL: <https://www.pinecone.io/learn/ai-hallucinations/> (visited on 05/29/2024).

REFERENCES

- [45] *Turing test*. en. Page Version ID: 1225753131. May 2024. URL: https://en.wikipedia.org/w/index.php?title=Turing_test&oldid=1225753131 (visited on 05/30/2024).
- [46] *FINN.no - Blink*. URL: <https://www.finn.no/blink> (visited on 05/02/2024).
- [47] *Hjem*. no. URL: <https://hjem.no/om-oss> (visited on 05/02/2024).
- [48] Thorvald Skaare Aschim. (+) – *Eiendomsmeglere er sulteforet på denne typen produkt*. nb-NO. Section: nyheter. Mar. 2024. URL: <https://www.shifter.no/nyheter/eiendomsmeglere-er-sulteforet-pa-denne-typen-produkt/316918> (visited on 04/30/2024).
- [49] *Prosper*. no-NO. URL: <https://www.prosper-ai.no> (visited on 06/05/2024).
- [50] *Smakfull 4-roms toppleilighet på Møllenberg | Baderom renoverert 22-TG0/1 | Kjøkken TG1 | Vindu 21 | Heis til p-plass*. nb-NO. URL: <https://www.finn.no/realestate/homes/ad.html?finnkode=344965597> (visited on 05/08/2024).
- [51] *COCO - Common Objects in Context*. URL: <https://cocodataset.org/#home> (visited on 12/21/2023).
- [52] *Datasets Arrow*. URL: https://huggingface.co/docs/datasets/about_arrow (visited on 05/08/2024).
- [53] Axel Thevenot. *Understand and Visualize Color Spaces to Improve Your Machine Learning and Deep Learning Models*. en. Feb. 2022. URL: <https://towardsdatascience.com/understand-and-visualize-color-spaces-to-improve-your-machine-learning-and-deep-learning-models-4ece80108526> (visited on 06/19/2024).
- [54] *sentence-transformers/all-MiniLM-L6-v2* · Hugging Face. Jan. 2024. URL: <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2> (visited on 06/02/2024).
- [55] *NbAiLab/nb-bert-base* · Hugging Face. Feb. 2024. URL: <https://huggingface.co/NbAiLab/nb-bert-base> (visited on 05/30/2024).
- [56] Lucas de Sá. *Text Clustering with K-Means*. en. Aug. 2020. URL: <https://medium.com/@lucasdesa/text-clustering-with-k-means-a039d84a941b> (visited on 06/19/2024).
- [57] Samet Oymak, Mingchen Li, and Mahdi Soltanolkotabi. “Generalization Guarantees for Neural Architecture Search with Train-Validation Split”. en. In: *Proceedings of the 38th International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, July 2021, pp. 8291–8301. URL: <https://proceedings.mlr.press/v139/oymak21a.html> (visited on 05/21/2024).
- [58] Asmida Ismail et al. “Improving Convolutional Neural Network (CNN) architecture (miniVGGNet) with Batch Normalization and Learning Rate Decay Factor for Image Classification”. en. In: *International Journal of Integrated Engineering* 11.4 (Sept. 2019). Number: 4. ISSN: 2600-7916. URL: <https://publisher.uthm.edu.my/ojs/index.php/ijie/article/view/4558> (visited on 11/24/2023).
- [59] Anil Johny and K. N. Madhusoodanan. “Dynamic Learning Rate in Deep CNN Model for Metastasis Detection and Classification of Histopathology Images”. en. In: *Computational and Mathematical Methods in Medicine* 2021 (Oct. 2021). Publisher: Hindawi, e5557168. ISSN: 1748-670X. DOI: 10.1155/2021/5557168. URL: <https://www.hindawi.com/journals/cmmm/2021/5557168/> (visited on 11/24/2023).

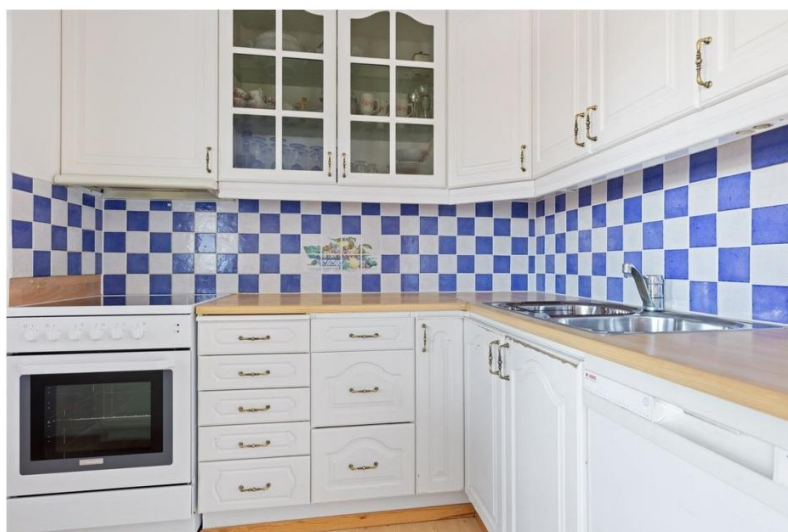
REFERENCES

- [60] *Papers with Code - Early Stopping Explained*. en. URL: <https://paperswithcode.com/method/early-stopping> (visited on 06/18/2024).
- [61] Bryan M. Li. *A batch too large: finding the batch size that fits on GPUs*. en. Oct. 2022. URL: <https://towardsdatascience.com/a-batch-too-large-finding-the-batch-size-that-fits-on-gpus-aef70902a9f1> (visited on 06/19/2024).
- [62] Samuel L. Smith et al. *Don't Decay the Learning Rate, Increase the Batch Size*. en. arXiv:1711.00489 [cs, stat]. Feb. 2018. URL: <http://arxiv.org/abs/1711.00489> (visited on 06/19/2024).
- [63] Sakshi Khanna. *A Comprehensive Guide to Train-Test-Validation Split in 2024*. en. Nov. 2023. URL: <https://www.analyticsvidhya.com/blog/2023/11/train-test-validation-split/> (visited on 05/21/2024).
- [64] *Salesforce/blip-image-captioning-base · Hugging Face*. May 2024. URL: <https://huggingface.co/Salesforce/blip-image-captioning-base> (visited on 06/02/2024).
- [65] *Salesforce/blip-image-captioning-large · Hugging Face*. May 2024. URL: <https://huggingface.co/Salesforce/blip-image-captioning-large> (visited on 06/20/2024).
- [66] *Salesforce/blip-image-captioning-base · getting different embedding values for the same image when trying to generate embeddings from two different sagemaker instances of the same model*. Oct. 2023. URL: <https://huggingface.co/Salesforce/blip-image-captioning-base/discussions/26> (visited on 05/26/2024).
- [67] *CC BY 4.0 Deed | Attribution 4.0 International | Creative Commons*. URL: <https://creativecommons.org/licenses/by/4.0/deed.en> (visited on 06/02/2024).
- [68] *Chi-squared test*. en. Page Version ID: 1228435745. June 2024. URL: https://en.wikipedia.org/w/index.php?title=Chi-squared_test&oldid=1228435745 (visited on 06/19/2024).
- [69] *How can I access GPT-4? | OpenAI Help Center*. en. URL: <https://help.openai.com/en/articles/7102672-how-can-i-access-gpt-4> (visited on 05/08/2024).
- [70] *Announcing GPT-4o in the API! - Announcements*. en. Section: Announcements. May 2024. URL: <https://community.openai.com/t/announcing-gpt-4o-in-the-api/744700> (visited on 06/05/2024).

Appendices

The following appendices are made to provide supplementary results and code snippets

Examples of Generated Captions



kjøkkenet har fliser over benk

Figure A.0.1: An example of a caption generated by model ID 14 (Descriptive)



kjøkkenet har godt med skap - og benkeplass som legger til rette for en god arbeidssone

Figure A.0.2: An example of a caption generated by model ID 15 (Creative)



benkeplaten i heltre med underlimt kjøkkenvask og fliser mellom benk og overskap

Figure A.0.3: An example of a caption generated by model ID 14 (Descriptive)



kjøkkenet er praktisk utformet med god skap - og benkeplass.

Figure A.0.4: An example of a caption generated by model ID 15 (Creative)



kjøkkenet er innredet med hvite, slette fronter, integrerte handtak, heltre benkeplate og integrerte hvitevarer

Figure A.0.5: An example of a caption generated by model ID 22



eldre kjøkken med enkel innredning

Figure A.0.6: An example of a caption generated by model ID 22



flott kjøkken fra nnnn med masse skap - og benkeplass

Figure A.0.7: An example of a caption generated by model ID 20



romslig kjøkken med gode vindusflater og mye naturlig lysinnslipp

Figure A.0.8: An example of a caption generated by model ID 22



kjøkkenøyen gir en sosial løsning og fin plass for spisebord

Figure A.0.9: An example of a caption generated by model ID 20



kjøkkenet er behagelig tilbaketrukket fra stuen

Figure A.0.10: An example of a caption generated by model ID 22



kjøkkenet har mye skap - og benkeplass som legger til rette for en god arbeidssone.

Figure A.0.11: An example of a caption generated by model ID 20



kjøkkenet er stilrent og smart utformet med en smart barløsning

Figure A.0.12: An example of a caption generated by model ID 22

Demonstration Videos of Website

Link to demonstrations video playlist

- https://www.youtube.com/playlist?list=PLt6fggMU3cCS29-mQvnmKB0ojG1Ubd_lk

The playlist contains:

Video with an introduction of the website:

- https://www.youtube.com/watch?v=JQjfDCFu2W8&list=PLt6fggMU3cCS29-mQvnmKB0ojG1Ubd_lk&index=1

Note that the following videos have added the model IDs in parentheses only for these video demonstration purposes:

A video comparing a realtor-written caption and BLIP models:

- https://www.youtube.com/watch?v=oboODXl1YCs&list=PLt6fggMU3cCS29-mQvnmKB0ojG1Ubd_lk&index=2&t=2s

A video comparing model ID 20, ID 18 (Large) and ID 22:

- https://www.youtube.com/watch?v=07ZflbzyTo4&list=PLt6fggMU3cCS29-mQvnmKB0ojG1Ubd_lk&index=3

A video comparing model ID 14 (Descriptive), and ID 15 (Creative):

- https://www.youtube.com/watch?v=cnLPkVnVL1Q&list=PLt6fggMU3cCS29-mQvnmKB0ojG1Ubd_lk&index=4

A video showing repetitions in generated captions and an example of poor data cleaning:

- https://www.youtube.com/watch?v=cnLPkVnVL1Q&list=PLt6fggMU3cCS29-mQvnmKB0ojG1Ubd_lk&index=4

Code Snippets for Automatic Evaluation

Average Sentence Length

```
def avg_sentence_length(column):
    text = " ".join(column)
    sentences = sent_tokenize(text)
    return sum(len(word_tokenize(sentence)) for sentence in sentences) /
        len(sentences) if len(sentences) > 0 else 0
```

Listing C.1: Code snippet for finding sentence length

TTR

```
def type_token_ratio(text):
    tokens = word_tokenize(text)
    types = set(tokens)
    return len(types) / len(tokens) if tokens else 0
```

Listing C.2: Code snippet for TTR

Yule's K

```
from collections import Counter
import math

def yule_k_measure(text):
    tokens = text.split()
    N = len(tokens)
    token_counts = Counter(tokens)
    V_m = Counter(token_counts.values())
    m_max = max(V_m)

    # Sum part calculation
    sum_part = sum([m**2 * V_m[m] for m in range(1, m_max + 1)]) / N**2

    # Yule's K calculation
    yules_k = 10**4 * (sum_part - 1/N)

    return yules_k * 10000 if i != 0 else 0
    return k
```


Listing C.3: Code snippet for Yule's K

BLEU

```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.translate.bleu_score import sentence_bleu, SmoothingFunction

def calculate_bleu(reference, candidate):
    smoothie = SmoothingFunction().method1
    return sentence_bleu(
        [word_tokenize(reference)],
        word_tokenize(candidate),
        weights=(0.25, 0.25, 0.25, 0.25),
        smoothing_function=smoothie
    )
```

Listing C.4: Code snippet for BLEU score

STS

```
from sentence_transformers import SentenceTransformer, util

model = SentenceTransformer('all-MiniLM-L6-v2')
gen_embeddings = model.encode(
    df['generated_caption'].tolist(),
    convert_to_tensor=True
)
true_embeddings = model.encode(
    df['true_caption'].tolist(),
    convert_to_tensor=True
)
cosine_scores = util.pytorch_cos_sim(
    gen_embeddings,
    true_embeddings
).diagonal()
sts_score = cosine_scores.mean().item()
```

Listing C.5: Code snippet for STS



 **NTNU**

Norwegian University of
Science and Technology