Wagon Number Localization Using a Vision-Language Model

IVAN MITEV, University of Twente, The Netherlands

This research explores the feasibility of using the Contrastive Language-Image Pretraining (CLIP) Vision-Language Model (VLM) for reading Unique Identification Codes (UICs) off of train wagons. The paper examines state of the art solutions to the problem of localizing wagon numbers on trains and tests CLIP's capabilities to be fine tuned for text localization. More specifically, it explains how four different fine-tuned versions of CLIP were trained and presents the results of their evaluation for the UIC localization. In the end, it is evaluated if CLIP shows promise in solving this task and should be pursued further or other methods are better suited.

Additional Key Words and Phrases: Computer Vision, Vision-Language Model, CLIP, Object Detection, Transformers

1 INTRODUCTION

With the large amount of trade between EU member-states, transporting goods in a fast and cost-effective manner has always been of key importance. Freight trains are one of the most prominent transportation methods, accounting for more than 360 billion tonne-kilometers per year [8]. Because freight wagons are often switched between trains, at different stations and different times, each wagon has a unique identifier code (UIC) [10] to keep track of it and its cargo. While there is a standardized UIC code structure, 11 digits, a dash and a check digit, formatting of the code and its placement on the wagon are open for interpretation. Figure 1 explains the structure of the code and presents one option for formatting it, the interoperability on the first line, the country code on the second and the rest of the code on a third line. On the other hand, Figure 2 presents an alternative format in which all of the digits are on the same line. This has led to the manual scanning and recording of UIC codes, costing countries and companies countless human-hours yearly [8]. The problem of efficiently scanning UIC codes, combined with the drive for a more sustainable industry has sparked research on an efficient and accurate method of solving the task. Many of the proposed solutions use AI models based on a residual network (ResNet) architecture, a variation of the Convolutional Neural Network (CNN) architecture [12][20]. Others are using a more hardware-heavy approach, for example, automated video gates [11]. However, one area of vision AI, vision-language models (VLMs)[5], has not been sufficiently explored. This paper will be concerned with testing if the Contrastive Language-Image Pretraining (CLIP) VLM model[18] has a competitive chance of solving the problem.

RQ: Do contrastive Visual Language Models offer a viable solution to the task of UIC code localization when compared with existing solutions?

SQ1: What are some state of the art solutions to reading wagon numbers?

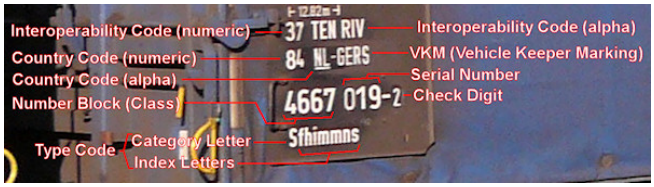

Fig. 1. UIC Format [10]

Fig. 2. Alternative UIC format

SQ2: What performance metrics can be used to evaluate the performance of a VLM model on UIC code localization tasks?

SQ3: Can CLIP's UIC code recognition capabilities be translated through different datasets with varying data.

This study will first examine already existing methods for reading wagon numbers in order to gauge the performance and efficiency required for a competitive alternative. Reading wagon numbers is a scene-text recognition (STR) task. This type of task is typically split into two sequential subtasks. First comes text localization which often includes drawing a bounding box around the edges of the text region. Then, the region is analyzed and the text is inferred from the shape of the characters [14]. This study will be mainly focused on the localization aspect. Multiple versions of CLIP will be trained on real-world data, provided by a Dutch railway company to classify between images containing UIC codes and images which don't. Each version will use the same pretrained base model but will be fine-tuned using a different dataset, varying in the number and type of images. After training, the fine-tuned versions will be tested both on data similar to the training data and data which is not.

The next chapter starts with a literature review, containing an explanation of VLMs and an overview of the inner workings of the CLIP mode. The third section discusses both state of the art solutions to wagon number reading and other fine-tuned versions of CLIP for scene-text recognition (STR). After that, the methodology of the experiment is described, complete with a description of the dataset, training method and how the model is evaluated. In the fifth section, the results are presented and in the sixth section the experiment is

TScIT 42, January 31, 2025, Enschede, The Netherlands

discussed as a whole. The last section is a conclusion, answering the research questions and suggesting directions for future research.

2 LITERATURE REVIEW

2.1 VLMs

Convolutional Neural Networks (CNNs) have been the cornerstone of deep learning for vision tasks, known for their ability to hierarchically learn features from images [18]. With advancements in hardware technology, Vision Transformers (ViTs) have gained prominence due to their capability to capture global context through self-attention mechanisms. However, ViTs are computationally expensive and energy-inefficient, posing sustainability challenges [6]. This high computational cost and energy consumption make ViTs less viable for widespread, environmentally sustainable applications.

Visual Language Models (VLMs) are a multimodal family of models, meaning they can learn from multiple types of inputs simultaneously, in this case images and text [5]. More specifically, these models take a visual and textual input and based on them generate a text output. They have shown very good generalization and zero-shot capabilities[15]. The four major families of VLMs include contrastive, masking, pretrained backbones and generative. In this paper, we will be focusing on the contrastive family, the recommended type when trying to teach a model representations which have a meaning both in the text and visual modalities[5]. In the case of UICs, these meaningful representations can be the different formats used for writing the codes. In simple terms, the contrastive approach trains a model with positive and negative image-text pairs. It teaches the model to associate similar images with the textual pairing of the positive example and disassociate them with text from the negative examples[5].

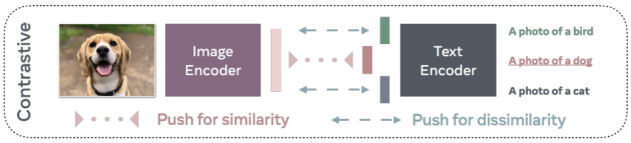

Fig. 3. Contrastive Model explanation

Transformers are a type of model architecture which focuses on keeping as much attention as possible. A transformer starts by turning its input into tokens by splitting it according to some parameter. Sentences might be tokenized into words by splitting the string on every sentence or into letters by splitting on every character. After a text is tokenized, each token is contextualized, meaning that it is made to store its relation to all other tokens in the input and this is how the token's tensor, also called a feature vector, is created[2]. This means that in a transformer model, the same word will have different representations, depending on where it is in the text and how it is used in a sentence. **Vision Transformers (ViT)** operate in a similar way, splitting an image into many segments and contextualizing each segment in the transformer's representation space[7]. Many versions of CLIP use this architecture[18].

Encoders are a part of the transformers architecture. They work by taking the input, splitting it into tokens and then transforming each token into a feature vector, also called tensor. There is exactly one tensor per token and each tensor is a contextualized representation of the token[1]. This means that it stores not only information about the token itself but also about its relation to all of the surrounding tokens in the input and is called self-attention[1]. The encoding of a simple textual input may look something like this: "From NYC to NYC" → {[0.1, 0.2], [0.1, 0.4], [0.3, 0.1], [0.2, 0.3]}. Each one-dimensional vector is a tensor representation of one of the words. These vectors are different, even when the words are the same, as they also store information not only about the contents of the word but also its context in the sentence, its position and relation to other words.

Decoders operate in much the same way as encoders, turning input into tensors. However, they operate on a masked self-attention principle, meaning that they carry context only for some of the surrounding tokens. While encoders are good at classification, question answering and understanding natural language, decoders are better at causal tasks such as generation[1].

A basic understanding of these concepts is required to understand how CLIP works. The CLIP version discussed in section 2.2 uses a transformer architecture with multimodal encoders, while the CLIP4STR modification discussed in section 3.2 also uses additional decoders.

2.2 CLIP

CLIP is a contrastive VLM which has many versions with different underlying architectures, but this study is concerned with a version, based on ViTs. This choice stems from ViTs' ability to generalize across different datasets and to make few and zero-shot predictions[6]. These qualities are important since some of the test data will differ from the training data in ways which are discussed in section 4.2. The model uses two separate encoders, one for images and one for text. After generating the tensors for the visual and textual tokens, it calculates their dot product and stores the result in a joint representation space, shown in Fig 4. This means that for N total images with N total text labels, the representation space will hold N*N products[16]. During training, the model tries to maximize the cosine similarity between the N correct pairings, represented by a blue diagonal in Fig 4. At the same time, it tries to minimize the similarity between the N*(N-2) incorrect pairings. A cosine similarity is a way of calculating similarity between two items, represented by vectors. By minimizing the cosine similarity, related objects move closer together on the representation space and unrelated objects move further away from each other. CLIP uses a symmetric loss function to minimize the similarity scores[18]. For the sake of simplicity, the only thing we need to know about symmetric loss is that when using it if the input changes by x units, the output also changes by x units. CLIP was trained in 8 different configurations, five of which are ResNets and three are ViTs. The model which is relevant for this paper is CLIP-ViT-B/32 which is a vision transformer which splits the input images into patches of 32x32 pixels. The model was trained on around 400 million (image, text) pairs taken from the internet [18]. After training, the model showed state of the art performance on zero-shot transfer, the task of being able to correctly classify previously unseen categories of images [3], for example classifying a new species as a type of beetle

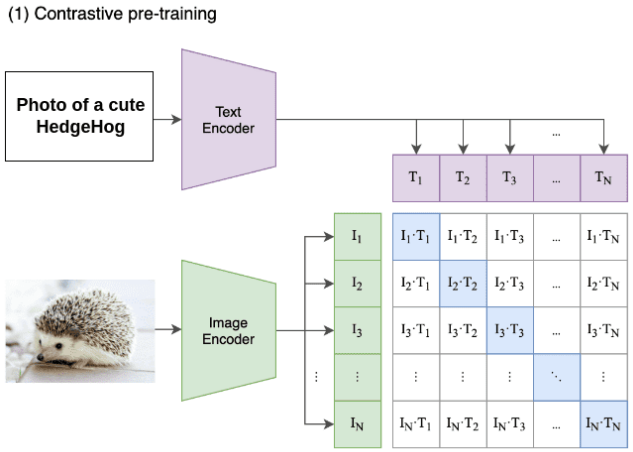

Fig. 4. CLIP's encoders

since it has the most in common with previously recorded beetles. On some datasets, it achieved an accuracy increase of more than 50%[18]. Other zero-shot experiments demonstrated that CLIP is able to correctly classify images without having seen them before based purely on text descriptions with accuracy up to 78.9% [17]. It also showed good scores on representation learning and robustness to natural shifts.

3 PREVIOUS WORK

3.1 Other Methods for Wagon Number Localization

3.1.1 CNN Approach. One approach to reading unique wagon codes is using a convolutional network (CNN). In short, CNNs leverage matrix multiplication to detect features in images, with each layer detecting goin more in-depth, starting from colours and edges and continuing with finer details. Ultimately, the image is converted into numerical values, through a feature extractor kernel„ enabling the CNN to extract all of those details [6]. This approach splits the code reading task into two portions: a character localization task and a character recognition task. The model is trained on a set of 5000 code patches with labelled ground truths for a total of 800 epochs. An ADAM optimizer is used with a learning rate of 1e-3, decaying one-tenth for every ten epochs, until a lower limit of 1e-5. In the end, another CNN is trained to check the last digit of the code, which is a check digit. After testing, the model performed with an accuracy of 93.98% and is able to process at a rate of 0.1s per frame [12]. It is important to note that the containers used in this research were under the chinese container numbering format which is different from the european UIC standard and includes letters as well as numbers.

3.1.2 ResNet Approach. Another approach used for counting wagons and efficiently reading wagon identification codes is utilizing a residual network (ResNet) based model [20]. Residual networks aim to address the problem, stemming from the addition of too many layers to a CNN in order to increase performance, which causes training progress to plateau. ResNets solve this by skipping blocks of layers in the CNN, turning them into a "residual" block [9]. This approach focuses on recognition from video data, analyzing the video frame by frame. It works in three stages: classifying if there is a wagon in the image, detecting a bounding box for the numbers and finally recognizing the numbers. A different ResNet model is used for each of the stages (ResNet-18, ResNet-34 and ResNet-50 respectively). Since the experiment uses video data, each wagon appears on multiple frames of the video, leading to high accuracies both on wagon counting and number recognition, 97.15% and 99.4% respectively. Another noteworthy feature of this approach is that it can process up to 11 frames per second even without utilizing a GPU[20].

3.2 CLIP4STR

Traditionally, scene text recognition (STR) has been a task, primarily handled by models which have a backbone trained on a single modality, despite multimodal models like CLIP being able to understand text[22]. Experiments showed that when CLIP was presented with an image containing text, unrelated to the contents of the image, it was often opted to select the text as a more probable label than the image itself[22]. This motivated researchers to develop a new version of CLIP which is able to retrieve text from images called CLIP4STR. This was done by adding an image decoder and a cross-modal decoder to the model's architecture and partially freezing the already existing text encoder. The resulting model first retrieves a visual prediction through the image encoder and then passes it to the image decoder which outputs the visual prediction. This prediction is used as an input to the partially frozen text encoder. Then, the output from the text encoder is combined with the output of the image encoder, concatenated and passed onto the cross-modal decoder which outputs the final text prediction. By utilizing the textual and visual modalities together, CLIP's text encoder is able to correct the predictions of the image decoder if they don't make sense. This is specifically useful in situations where text is occluded. Different versions of CLIP4STR were then trained with anywhere between 155 million to 1 billion parameters and 3.3 million - 6.5 million training data examples. After training, CLIP4STR was tested on 10 common benchmarks and achieved state-of-the art results when compared to other str models such as PARSeq and ViTSTR-S[22].

3.3 Object localization using CLIP

There have been multiple studies on fine-tuning CLIP for object localization. This task is very similar to UIC localization, as long as the model is able to learn thoroughly enough the concept of a UIC. One of the models which has best implemented object detection using CLIP is OWL-ViT [13]. It uses CLIP as the base model in order to leverage its extensive knowledge of different text labels and all of their possible image representations. However, the architecture of the image encoder is altered to turn it from a classification model to a detection one. Also, the text encoder is altered to feed text tokens into the image encoder. By doing this, it is possible to convert the model for Open Vocabulary Object Detection, a version of object detection, where the model can predict classes, outside of the based ones used during training [13]. Both of the modified encoders are

then trained from scratch on a dataset of 3.6 billion image-text pairs and are then further fine-tuned on a dataset of 2 million images. After testing, the model proved to achieve state-of- the art zero-shot object detection on the LVIS dataset [13].

4 METHODOLOGY

The CLIP version that was chosen for training was CLIP-ViT-base-patch32. This was mainly done because Res-Net like architectures have already been explored for wagon number reading[20] and patch32 is the best-performing classifier out of the CLIP ViT models[18]. Also, a smaller model is more computationally efficient to fine-tune as less parameters need to be stored in memory. This model was then fine-tuned for the task of binary image classification, to determine whether or not an image contains a UIC code.

4.1 Experiment Schema

Figure 5 shows the experiment schema and how the data was processed, used for training and also for testing.

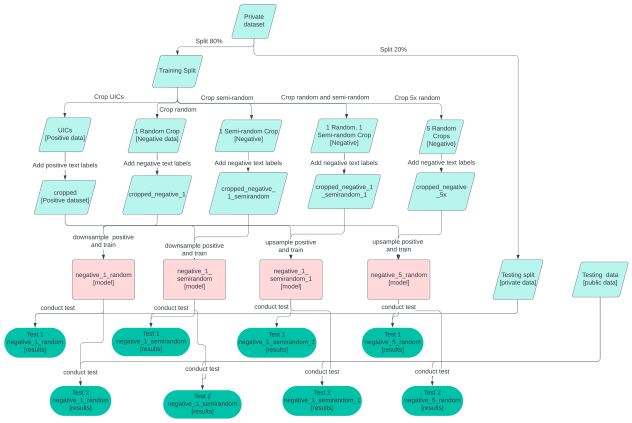

Fig. 5. Experiment Schema

4.2 Data

The data used for fine-tuning was a set of labeled images, taken with a line-scan camera, provided by a railway company. Line-scan cameras are often used to take images of moving trains, producing a high-quality picture of the train, where static parts of the photo such as the background appear blurred. The dataset contained 8283 images with a height of 2048 pixels and width from 6536 to 12846 pixels. The images were all taken from the same distance with the same camera but on different days and hours, causing a variety of lightning conditions. The labels, relevant to this experiment, were the coordinates of a bounding box around the location of UIC codes and the transcribed UIC codes themselves.To be used as training data, the images and labels needed some preprocessing. While all of the images contained train wagons, some data sanitization was required. First of all, not all of the images contained UIC codes and some of them contained multiple UIC codes, leading to a very large bounding box, spanning multiple wagons. These images were pruned from the dataset to ensure uniformity in the training data.

The remaining dataset was split into parts of 80 and 20 percent to create the training and testing sets. Then, the UIC codes were cropped from the private dataset, according to the bounding box labels, and from each image a batch of completely and semi-random images with the same size as the UIC bounding box were also cropped. Additionally, an intersection test was conducted in order to ensure that the crops didn't include any part of the area within the bounding box. To ensure variety and sufficient difference between the training data in the multiple versions of the model, described in the next section, the random and semi-random crops were combined in the five datasets, described in table 1. The "cropped" dataset includes the cropped UIC bounding boxes. The "cropped-negative-1" includes 1 random crops from the original photos. One crop per original photo. The images are less than the "cropped" dataset because some crops didn't meet the no-overlap with bounding box condition. The "cropped-negative-1-semirandom" dataset includes random crops from a distance of (50+n) pixels from the bounding box, where n is the largest dimension of the box. These types of crops will be referred to as semi-random. The "cropped-negative-1-semirandom-1" includes a combination of random and semi-random crops as described in the previous two datasets with two crops per original photo. Finally the "cropped-negative-5x" includes completely random crops from the original images with five crops per image.

Dataset	Number of images
cropped	6626
cropped-negative-1	6607
cropped-negative-1-semirandom	6405
cropped-negative-1-semirandom-1	13015
cropped-negative-5x	33035

Table 1. The training datasets

The models will be tested on two datasets, labeled as "Testing split" and "Testing data" in Figure 5. One of them is the 20 percent of the private dataset which was not used for training. This dataset contains about 1800 images. The other is a public wagon dataset with around 750 labeled images. This public dataset contains pairs of jpg-json files, with the json file containing labeled bounding boxes and transcriptions for all UIC or tonnage signs found on the image, if any. The images were taken with a handheld camera and therefore have a lesser quality than the line-scan images, at 1024x768 pixels. Unlike the line-scan images, they vary in distance from the wagon, angle, blurriness and type of wagon being photographed. This means that compared to the private dataset, the public one has a wider range of both the types of images and the UIC placement and formatting.

4.3 Model Training

The model chosen for fine-tuning was CLIP's ViT-B/32 version, due to the fact that it shows state of the art performance on classification tasks [18] and the larger patches will result in faster training time. To solve the task of UIC localization, four different versions of the model were developed to distinguish between images containing a

[h]

Model "negative-"	Precision	Recall	Accuracy	F1	Loss
1-random	0.990	0.939	0.964	0.964	0.099
1-semirandom	0.972	0.947	0.960	0.959	0.158
1-semirandom-1	0.988	0.928	0.958	0.957	0.117
-5x	0.989	0.978	0.984	0.984	0.044

Table 2. Validation metrics

UIC and images which don't contain one [14]. The "cropped" dataset was used as the positive class and all images in it were labeled "An image of a UIC code", while the images in all of the other datasets were classified as negative and labeled "An image which doesn't contain a UIC code". Each version used only one of the "cropped_-negative" datasets as negative data and therefore we can name the versions after their negative datasets. Training four versions on different types of clearly defined data makes it easier to determine what type of data can successfully be used to fine-tune the model for this task.

During training, the standard 80/20 split into training and validation data was performed. The models are all trained with batch sizes of 32, learning rates of 5e-5[4] and weight decay of 0.1. Also, an AdamW optimizer was used to implement a learning rate scheduler which reduced the learning rate decay by a factor of 0.5 when f1 score plateaued for 2 epochs or more. All of the versions were trained for 10 epochs, except the "cropped_negative_5x" version which was trained for 7 epochs due to the higher number of data-points.

4.4 Model Evaluation

During training, the models recorded the metrics of precision, recall, f1 score, accuracy and loss for each epoch. Table 2 shows the values of these metrics after the final epoch.

Precision is a measure of the total number of true positives, divided by the sum of true positives and false positives and represents and reflects the ability of a model to correctly predict positives.Recall is the number of true positives, divided by the sum of true positives and false negatives and reflects how often the model correctly identifies positive instances. Accuracy is all of the correct guesses divided by the sum of all guesses and reflects how often the model makes correct predictions, no matter if negative or positive. The F1 score is two times the precision multiplied by the recall and divided by the sum of the precision and the recall and again is a measure of how often the model is expected to make a correct prediction [19]. Loss is a metric that computes how far off predictions are from the actual value in the model's representation space.

To evaluate the performance of the fine-tuned models, two tests were developed. The tests were then performed on two datasets. The 20 percent part of the images which was not used for cropping and a publicly available dataset of train wagon photos.

Firstly, a UIC detection test is performed using a sliding window protocol. A bounding box region of NxM pixels, shaped like a rectangle, is created at the corner of the image and is then moved in steps. For the private dataset, the window was 500x400 pixels with a step of 50 pixels because the images are of very high pixel rate and sometimes the uic codes are on multiple lines. For the public dataset, the sliding window was 500x200 pixels in size with a step of 40 pixels as the images are lower quality but are taken from closer distances and they are more often on a single line. On each step, binary classification is performed with the same prompts as those used in training the models and the probability scores for the positive and negative labels are recorded. In the end, the bounding box with the highest confidence level and the bounding box with the highest Intersection over Union (IoU) are selected if the image contains a UIC label. The IoU is a measure of how much two bounding boxes in an image overlap and is defined as the area of overlap divided by the area of their union. The coordinates, confidence intervals and IoUs of the two windows are recorded.

These results are then converted to a confusion matrix, which doesn't have a true negative. The decision to not record true negatives is intentional as with a sliding window protocol, there would be way more true negatives than any other results as most of the sliding windows would not contain a UIC code in them. If the two bounding boxes match, the result is counted as a true positive. If the boxes don't match and the confidence interval of the window with highest IoU is less than 50%, the result is a false negative. If both of the windows have a confidence interval of more than 50% but their IoUs with the original bounding box differ by more than 10%, this suggest that only a part of the UIC is included in one of the windows and it has wrongly categorized the area as containing a code, so it is a False Positive. If both windows have a confidence above 50% but their IoUs differ by less than 10%, this suggests that the missing part of the bounding box is not so significant as to harm the model's inference capabilities and the result is still counted. After recording the results, precision, recall and f1-score will be calculated.

The second test examines successful classification of the labeled UIC regions. Here, inference will be run on the cropped areas within the UIC regions. The datasets used will be the 20% split from the private dataset which was not used in training and the public dataset. For each image within the bounding box, the models will also be asked to classify one randomly cropped image, taken from the same original photo, with the same dimensions as the bounding box but which does not overlap with it. Again, a confusion matrix will be Here, every correct classification of a cropped UIC region will be counted as a true positive, every incorrect classification as a false negative. Also, every correct classification of the randomly cropped area will be counted as a true negative and every incorrect classification as a false positive. After recording the results, precision, recall, accuracy and f1-score will be calculated.

5 RESULTS AND DISCUSSION

5.1 Test 1 Results

Test 1 was conducted successfully as described in section 4.4 and its results are presented in tables 3 and 4. Figures 6 and 7 are examples of the visualized results from the public dataset. Note that Figure 7 is still an example of a True Positive as there are two sliding windows with an equally large IoU with the bounding box. In the end, 405 files were examined as a portion of the images didn't contain UIC codes. The model "negative_1_random" had a pretty even distribution of positives and negatives but they were made randomly and resulted in more than half of the predictions being false negatives. On the other

hand, "negative_1_semirandom" tended to over-predict for the negative, resulting in over 85% of the positive predictions being True Positives. The other two had zero false negatives, but a precision score which suggests they were predicting positively during the whole test.

With the private dataset, the test was only conducted on 71 images, as the processing time per image was in magnitudes larger than with the public dataset. This happened because the images were larger, of higher quality and taken from further away, resulting in a relatively small sliding window which had to perform a lot of steps. On this dataset, "negative_1_random", outputted 74% false negatives and 26% false positives without any true positives, resulting in no precision, recall and F1 values. "Negative_1_semirandom" had a single true positive, while the other two models predicted only false positives.

Model	Precision	Recall	F1
negative-1-random	0.365	0.263	0.305
negative-1-semirandom	0.877	0.126	0.220
negative-1-semirandom-1	0.393	1.000	0.564
negative-5x	0.304	1.000	0.466

Table 3. Test 1 Public

Model	Precision	Recall	F1
negative-1-random	0.000	0.000	0.000
negative-1-semirandom	0.125	0.016	0.028
negative-1-semirandom-1	0.000	0.000	0.000
negative-5x	0.000	0.000	0.000

Table 4. Test 1 Private

Fig. 6. Test 1 example of a False Positive

Fig. 7. Test 1 example of a True Positive

5.2 Test 1 Discussion

The discrepancy in the results between the two datasets suggest that the model gained some understanding of how to interpret the public data correctly but didn't do the same for the private data. However, several factors can disprove this theory. First of all, the private data much more closely matches the validation data on which the model achieved correct predictions. As the private testing data and the validation data are a subset of the same dataset, the fact that the models achieved better performance on a more varied dataset raises some alarms. Secondly, the images in the private dataset are a lot larger and thus require more sliding window steps to be processed. As we know, the models tend to over predict the presence of a UIC code. On the smaller, public images this partially random prediction can yield them successful lucky guesses. On top of that, the public dataset had more images, raising the total number of attempts the model has to make a lucky guess during the test. Last but not least, the work done on OWL-ViT suggests that even with architectural modifications, CLIP needs to be fine-tuned on millions of labeled images to successfully detect objects [13].

5.3 Test 2 Results

In the second test, all models except "negative_1_semirandom" predicted only positive values. "negative_1_semirandom" also showed a heavy bias for predicting positive values but sometimes predicted negative values as well. Also, in more than 85% of the times it predicted a negative value, the value was actually negative. In this test, the number of positive and negative examples for the public and the private datasets were not equal but were within 7% of each other. These results are presented in tables 5 and 6, while figures 8 and 9 show examples of two crops from the public dataset.

5.4 Test 2 Discussion.

The results from test 2, combined with the validation metrics from the last epochs of training show that the models overfit the dataset. The models have learned to correctly classify their training data

Model	Accuracy	Precision	Recall	F1
negative-1-random	0.503	0.000	1.000	0.000
negative-1-semirandom	0.534	0.502	0.973	0.662
negative-1-semirandom-1	0.503	0.503	1.000	0.669
negative-5x	0.503	0.503	1.000	0.669

Table 5. Test 2 Public

Model	Accuracy	Precision	Recall	F1
negative-1-random	0.487	0.487	1.000	0.655
negative-1-semirandom	0.531	0.509	1.000	0.675
negative-1-semirandom-1	0.487	0.487	1.000	0.655
negative-5x	0.487	0.487	1.000	0.655

Table 6. Test 2 Private

Fig. 8. Test 2 Positive Example

Fig. 9. Test 2 Negative Example

but have not learned how to identify images containing a UIC. A possible explanation, as mentioned in the discussion of test 1 might be that the model was not trained on enough data. However, the prevailing positive classifications show that the negative data used in training was not varied enough and was too different from the positive examples. This is further supported by the fact that the semi-random model achieved the best performance, since the negative data in this dataset is most likely to include characters, digits and other closely related UIC pictures. The idea of training the model on negative examples closely related to the positive data is supported by research and is referred to as "hard negative examples" [21]. Also, the models demonstrated similar on both datasets, even showing marginally better performance on the public dataset. This suggests that their performance was not dependent on the variety of test data, but on the training methodology.

6 CONCLUSION

This paper explored the visual-language model CLIP and its ability to detect UIC codes on train wagons. It discussed state of the art solutions based on different architectures such as ResNet and CNNs which have achieved accuracies of up to 99% . Based on the experiments made during this research and the explored literature, it can be inferred that CLIP shows potential for UIC localization. The research demonstrated that traditional performance metrics such as IoU can be used to effectively evaluate CLIP for code localization. It also showed that CLIP's performance can be successfully translated across varying datasets as long as the model has been properly trained. Based on the findings of this paper, future work should focus on determining clear criteria for data selection when fine-tuning CLIP such as the amount of data and the variance between positive and negative examples

ACKNOWLEDGMENTS

I would like to extend my gratitude towards Melissa Tijink for providing her guidance, technical expertise and invaluable dataset, and to both Melisa and Shunxin Wang for their advice and patience during our progress meetings. I would also like to thank Luuk Spreeuwers for his contribution and analysis in the final weeks of the research.

REFERENCES

[1] Kyle Aitken, Vinay V Ramasesh, Yuan Cao, and Niru Maheswaranathan. 2021. Understanding How Encoder-Decoder Architectures Attend. arXiv:2110.15253 [cs.LG] https://arxiv.org/abs/2110.15253
[2] Amanatullah. 2023. Transformer architecture explained. https://medium.com/@amanatulla1606/transformer-architecture-explained-2c49e2257b4c
[3] Dave Bergmann. 2024. What is zero-shot learning? https://www.ibm.com/think/topics/zero-shot-learning
[4] Vrunda Bhattbhatt. 2024. Learning rate and its strategies in neural network training. https://medium.com/thedeephub/learning-rate-and-its-strategies-in-neural-network-training-270a91ea0e5c#:~:text=The%20choice%20of%20learning%20rate,become%20stuck%20in%20local%20minima
[5] Florian Bordes, Richard Yuanzhe Pang, Anurag Ajay, Alexander C. Li, Adrien Bardes, Suzanne Petryk, Oscar Mañas, Zhiqiu Lin, Anas Mahmoud, Bargav Jayaraman, Mark Ibrahim, Melissa Hall, Yunyang Xiong, Jonathan Lebensold, Candace Ross, Srihari Jayakumar, Chuan Guo, Diane Bouchacourt, Haider Al-Tahan, Karthik Padthe, Vasu Sharma, Hu Xu, Xiaoqing Ellen Tan, Megan Richards, Samuel Lavoie, Pietro Astolfi, Reyhane Askari Hemmat, Jun Chen, Kushal Tirumala, Rim Assouel, Mazda Moayeri, Arjang Talattof, Kamalika Chaudhuri, Zechun Liu, Xilun Chen, Quentin Garrido, Karen Ullrich, Aishwarya Agrawal, Kate Saenko, Asli Celikyilmaz, and Vikas Chandra. 2024. An Introduction to Vision-Language Modeling. arXiv:2405.17247 [cs.LG] https://arxiv.org/abs/2405.17247
[6] Luca Deininger, Bernhard Stimpel, Anil Yuce, Samaneh Abbasi-Sureshjani, Simon Schönenberger, Paolo Ocampo, Konstanty Korski, and Fabien Gaire. 2022. A comparative study between vision transformers and CNNs in digital pathology. arXiv:2206.00389 [eess.IV] https://arxiv.org/abs/2206.00389
[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv:2010.11929 [cs.CV] https://arxiv.org/abs/2010.11929
[8] European Rail Freight Association (ERFA). 2022. The European Rail Freight Market Competitive Analysis and Recommendations. https://www.erfarail.eu/uploads/The%20European%20Rail%20Freight%20Market%20-%20Competitive%20Analysis%20and%20Recommendations-1649762289.pdf
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385 [cs.CV] https://arxiv.org/abs/1512.03385
[10] Aaron Hoore. 2009. https://nl.dbcargo.com/resource/blob/1430008/9767e97bb070ccbbf77efd84e7d64948/freight_wagon_catalog_v2011-data.pdf
[11] Behzad Kordnejad, Martin Kjellin, Martin Aronsson, Guillermo Garcia, Santiago Vilabella, Rico Wohlrath, Ingrid Nordmark, Roald Lengu, Mats Åkerfeldt, and Jan Bergstrand. 2022. Intelligent Video Gate -Automated Detection of Wagons and Intermodal Loading Units for Image Processing and Sharing and Exploitation of Data. (06 2022).
[12] Chenghao Li, Shuang Liu, Qiaoyang Xia, Hui Wang, and Haoyao Chen. 2019. Automatic Container Code Localization and Recognition via an Efficient Code Detector and Sequence Recognition. In *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM).* 532–537. https://doi.org/10.1109/AIM.2019.8868819
[13] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, and Neil

Houlsby. 2022. Simple Open-Vocabulary Object Detection with Vision Transformers. arXiv:2205.06230 [cs.CV] https://arxiv.org/abs/2205.06230
[14] Lukáš Neumann and Jiří Matas. 2015. Efficient Scene Text Localization and Recognition with Local Character Refinement. arXiv:1504.03522 [cs.CV] https://arxiv.org/abs/1504.03522
[15] Merve Noyan and Edward Beeching. 2024. Vision language models explained. https://huggingface.co/blog/vlms
[16] Szymon Palucha. 2024. Understanding openai's clip model. https://medium.com/@paluchasz/understanding-openais-clip-model-6b52bade3fa3
[17] Qi Qian and Juhua Hu. 2024. Online Zero-Shot Classification with CLIP. arXiv:2408.13320 [cs.CV] https://arxiv.org/abs/2408.13320
[18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020 [cs.CV] https://arxiv.org/abs/2103.00020
[19] Koo Ping Shung. 2020. Accuracy, precision, recall or F1? https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9
[20] Andrey Vavilin, Andrey Lomov, and Titkov Roman. 2022. Real-time Train Wagon Counting and Number Recognition Algorithm. In *2022 International Workshop on Intelligent Systems (IWIS)*. 1–4. https://doi.org/10.1109/IWIS56333.2022.9920835
[21] Hong Xuan, Abby Stylianou, Xiaotong Liu, and Robert Pless. 2021. Hard negative examples are hard, but useful. arXiv:2007.12749 [cs.CV] https://arxiv.org/abs/2007.12749
[22] Shuai Zhao, Ruijie Quan, Linchao Zhu, and Yi Yang. 2024. CLIP4STR: A Simple Baseline for Scene Text Recognition With Pre-Trained Vision-Language Model. *IEEE Transactions on Image Processing* 33 (2024), 6893–6904. https://doi.org/10.1109/tip.2024.3512354

A AI DISCLOSURE

During the writing of this thesis, the author used the AI chatbot Claude to suggest ideas and troubleshoot coding errors. After using this tool, the author reviewed and edited the responses as needed and takes full responsibility for the content of the work