UNIVERSITETET I AGDER

MASTER'S THESIS

A DEEP LEARNING SEGMENTATION APPROACH FOR AUTOMATIC INVOICE IDENTIFICATION

IKT590

WRITTEN BY:
ELLEN SYNNØVE SAMDAL

SUPERVISORS:
MORTEN GOODWIN
JAHN THOMAS FIDJE

FACULTY OF ENGINEERING AND SCIENCE
UNIVERSITY OF AGDER

GRIMSTAD

24.05.2019

STATUS: FINAL

Abstract

The research on object detection has come a long way and has in recent years become so efficient and reliable that we see many practical applications in simple detection of objects in pictures to advance segmentation in video recordings. Despite its popularity, there are no de facto approach available and several methods are competing to be superior.

This thesis focuses on object detection in photos where the desired object is letters and numbers, and the photo is a PDF-file of an invoice without any use of OCR and any knowledge of the layout. The research compares Tensorflow's Object Detection API with SSD Inception V2 and YOLO. One of the methods shows promise, though it will need some more work. This thesis shows that SSD seems to need more training, possibly with more varied data, while YOLO seems to not be a good method for this type of object detection.

The results from SSD varies as it sometimes get the correct location, though not always being certain in its decision. For other invoices it misses completely, and might guess at other short words or numbers. Sometimes the words and numbers make sense, and could be the start of understanding what it is looking for. The tests done with YOLO gives results where it guesses almost the same location for every invoice, regardless of where the object actually is. It seems to overfit on the most popular formats, which is going against what this thesis is trying to do.

Keywords: Invoice, Object Detection, SSD, YOLO

Abbreviations

Abbreviation	Explanation
OCR	Optical Character Recognition
NN	Neural Network
CNN	Convolutional Neural Network
SSD	Single Shot Detection
YOLO	You Only Look Once
R-CNN	Region-Convolutional Neural Network
IoU	Intersection over Union

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Definition

We humans always want to make life easier in every way possible, and with machine learning we are creating solutions that do this. One small part of life that can be annoying is paying an invoice; typing everything in can be a chore when long numbers are involved. This thesis focuses on object detection of invoices, particularly of invoice identification number. If this is possible, then the rest of the invoice should be possible, and if this is done, then other invoices could be processed automatically no matter the layout. This could also be translated to other types of documents with fields of text.

This problem is hard, as there are many "objects" that look the same in an invoice, as well as the data being in a different order from invoice to invoice. A date could be at the top on one invoice while the business name could be on the top of another, which could confuse a learning system. In addition, some invoice identification numbers could be of a different size than the normal text, which would be easier for people to see, but might be unexpected for the model, especially as the font is somewhat different too. This could possibly be closer to a company name or logo, which could be confusing for the model.

The two methods that are tested in this thesis are Tensorflow's Object Detection API with SSD Inception V2 and YOLO. Both of these work well with regular object detection, and this thesis tests whether this can be translated to not so regular objects. The goal of this is to build a model where the layout of the document does not need to be known as long as the model knows what classes could be found there, and where using Optical Character Recognition (OCR) to "read" the document to find hints of where the objects are is not needed either.

1.1.1 Research Questions

- Can an invoice identification number be recognized without prior knowledge of the format used?
- Is it possible to find an object within a document without OCR?

1.2 Report Outline

This report will present the background information needed to understand the problem and the experiments in chapter 2, in addition to the state-of-the-art at the time of writing. Chapter 3 will explain what kind of methods were tested and how they were tested. The results from these experiments will be discussed and evaluated in chapter 4, to try to understand what has happened in the model and why. This will include some examples of the output. The thesis will be summarized in chapter 5, and finally there will be some proposed ways of taking this further in chapter 6.

Chapter 2

Background Theory

This chapter will explain the necessary theory to understand this thesis and its experiments. Section 2.1 explains how different neural networks work, while section 2.2 shows how a single shot multi-box detector works. YOLO, what it is and how it works, will be discussed in section 2.3. The final section in this chapter, 2.4, presents the state-of-the-art for invoice recognition and auto filling.

2.1 Neural Networks

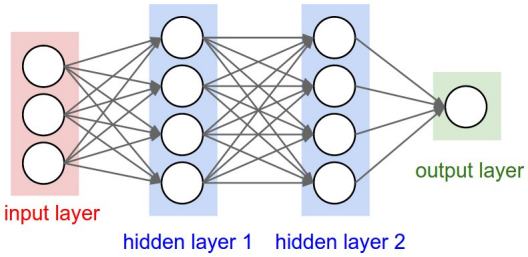


Figure 2.1: Regular NN [1]

A neural network (NN) is made up of several layers of neurons. Those neurons work by receiving several inputs, and giving a single output based on the inputs and their respective weights (the importance of the input). This output will be used as input to the next layer, and finally the network gives a final output as the predicted answer. In object detection the inputs would be the pixels in the picture, and the final output would be the predicted object. To the right is figure 2.1 showing an example of a regular NN. [2]

The problem with using regular neural networks with images is that it does not scale well at all. The reason for this is that all the neurons are connected and need their own weights, and the more pixels the image has, the more weights have to be saved. CNN is more reasonable to use when working with images because the neurons in one layer is only connected to some of the previous layer instead of all of it. The neurons are also arranged in three dimensions: width, height, and depth. Width and height are the input image's size, while depth represent the number of channels. Depth is usually three, one for each of the colors red, green, and blue. Figure 2.2 to the left shows a convolutional NN. [1]

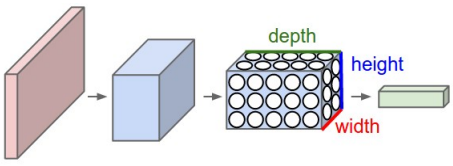


Figure 2.2: Convolutional NN [1]

2.2 SSD

The single shot multibox detector's architecture is built on VGG-16's (a CNN for detection and classification). Single shot means that the detection and classification is done on a single forward pass of the network, and multibox is a technique for bounding box regression. Detector means that the network is a detection network, and it also classifies the detected objects. It consists of two parts: extract feature map and apply convolution filters to detect objects. VGG-16 is used to get the feature map, then it uses a convolutional layer to predict objects. This layer is divided into 38x38 cells, and it predicts an object four times per cell. The prediction that scored best is the class it tells us is there. One of these classes is "nothing", so that the areas of the image that contain nothing of interest do not show any predictions. [3]

2.3 YOLO

YOLO, You Only Look Once, is another method of predicting several objects in a single image. The input image is divided into S x S cells. For each of the objects that are present in the image, the cell where the center of the object's bounding box is located is set as "responsible" for the prediction of the object. Each of the cells predict bounding boxes and classes, and the coordinates are normalized to be between 0 and 1 [4]. The cells only predict two boxes and can only have one class, which means that if there are several small objects close together, some of them might not get predicted [5].

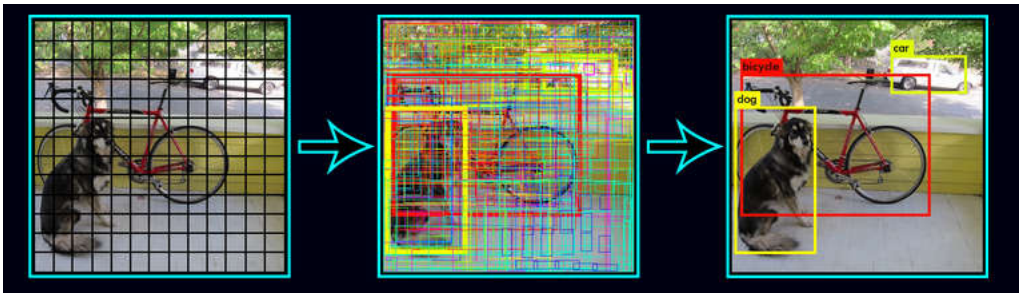


Figure 2.3: How YOLO works [6]

2.4 State-of-the-Art

There are many possible methods to find objects in images. The best method as of May 2019 for object detection is one called TridentNet [7]. What sets this apart from other object detection methods is how it handles scale variation [8]. We have one stage methods, methods that predict bounding boxes in one go, and two stage methods, which first generate areas where an object may be, and then predicts what objects are where. Both of these types of methods generally have a problem with scale variation, which TridentNet solves.

One of the best two stage methods is Faster R-CNN. We have R-CNN, Fast R-CNN and Faster R-CNN. As the names suggest, Fast R-CNN is faster than R-CNN, and Faster R-CNN is the fastest of the three methods. When ignoring the time it takes to propose the regions, Fast R-CNN "achieves nearly real-time rates" [9].

YOLO is one of the most popular methods for object detection. It is a one stage method, and is famous for being extremely fast, so fast that it can classify in real time [5]. When speed is important, YOLO is one of the best methods.

Another one stage method that is popular is SSD. It is slower than YOLO, but faster than Faster R-CNN, and the accuracy is also between these two methods, though the larger the object the closer it gets to Faster R-CNN's accuracy [10]. This method can also be run in real time. Most people will see these two methods as the most viable ones when classifying videos.

When it comes to invoice data extraction, there are two methods. The first is to match the invoice to a known layout. This is a safe method, but it takes a lot of time and energy to set up the rules for each different invoice layout, meaning that when the goal is to be able to "read" any invoice this is not ideal [11]. If the goal is rather to automatically pay invoices from a few different businesses, then template matching is absolutely the best solution.

The other method of invoice data extraction is one similar to what this thesis is about. It is either to find a number or word and then search in the near vicinity for a field name to match it to, or to search for the field name first and then find the corresponding number or word [12], [13]. The difference between this and what this thesis wants to achieve is that the goal of this thesis is to not need any prior knowledge, whereas these solutions require the knowledge of what field names to assosiate different values with.

At the moment, at least two Norwegian banks give you the option of taking a picture of your invoice, scanning it, and filling it out for you [14], [15]. They have not published what kind of method they use, so it is hard to know if it uses OCR to find the appropriate fields, or if it manages to find the fields and then use OCR afterwards to fill it in. It is also possible that they find out what kind of format the invoice has and know where the fields are based on that, but this is unlikely as that would limit the product severely, or take a lot of time to train to recognize countless formats, and the instruction to double check every field is more likely because because the product can find the wrong fields, not because it cannot recognize the format.

Chapter 3

Methods

OCR will not be used to find the invoice id, as this requires the format to be known. What we want is a system that works regardless of format, so that it can be used on any invoice without prior knowledge of how it looks. There are two methods that were used to try to find the invoice identification number, which will be explained in this chapter.

All the images in this chapter and the next are examples of the training data and the output from the trained models. All the information is blurred, as these invoices are real, and should not be public. The exception is the invoice identification number, so that everyone can see clearly both the boxes and where the boxes actually should be. The field names (for instance "Customer Number" (or "Kundenummer" in Norwegian)) are also not blurred, as this is not personal information, and will only help with understanding what kind of data the models have to train with. The invoices have been cut to conserve space, but they are accessible in full in Appendix A. All of the images are of the same corner, the top right one, unless stated otherwise.

3.1 Prework

The data was sent as a little over 3000 PDF files with one accompanying json file. The json file contained all the fields in the PDFs, which were all invoices. To be able to work with the invoices they first had to be converted to an image type file. This was the first obstacle. To convert from PDF to png (the desired file type) a DPI was needed which was not included. This meant that some time was wasted with trial and error to find the correct DPI. There was one DPI that worked for most of the invoices, but it was not perfect for absolutely all. This means that some on the training data has a slightly wrong location, but this should not be a problem as it is likely to be very few of them.

Figure 3.1: How tight most of the boxes are and should be

One such example can be seen in figure 3.4 on the next page, but this could also be just an error in the json file. In this example the invoice identification number is marked with "Invoice ID number" in red. An example of how the box should look can be seen in figure 3.1. There are other errors in the file, but these are mostly errors that make the bounding box larger than necessary, as in figure 3.5, smaller than what it should be, resulting in not the whole number being included like in figure 3.6, or slightly off, as we can see in figure 3.2. The invoices with the whole box completely missing could also be an error of the same kind (error in the json file) instead of getting the wrong DPI when converting, as this was not tested. Figure 3.3 is an example of a box where it is hard to see at first glance whether it is wrong DPI or human error, as the box is quite snug around the day and month of the date. These errors were not corrected, in the hope that it would decrease the chance of overfitting.

Figure 3.2: Not completely right

Figure 3.3: A miss, or human error?

Figure 3.4: Possibly wrong DPI in convertion

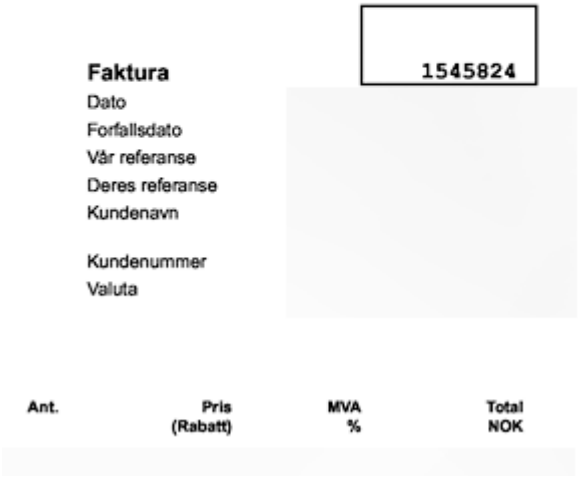


Figure 3.5: A box with a lot of extra space

Figure 3.6: Too small box, not every number is included

The next objective was to split the json file into seperate files. For the SSD tests the files had to be of type csv, with one file for training and one for testing. The split was 90% for training and 10% for testing, which meant that the last 78 of the invoices had to be excluded from the two csv files. These were put in their own file, but this was not necessary as this was never used. The YOLO tests, however, had to have the files as type xml, and with one image having its own xml file. As the original json file contained all of the fields, most of it was useless. When going through the json file there was a search on "invoiceid" to find all the information needed for the location of the invoice identification number. This information is what was saved in the csv files, and later the xml files. Another small obstacle was that the file names saved in the json file, which were meant to point to which invoice the information was for, had replaced underscores with dashes. This did not become apparent before trying to convert from PDF to png, as the method of testing the DPI was to paint a box where the json file said the invoice identification number was. This, of course, tried to open the files which were in the csv file, and these were not found. After going through all the entries in the csv files and converting from PDF to png, the images and annotations were ready to be trained on.

3.2 SSD

The first method tested was Tensorflow's Object Detection API with the SSD Inception V2 model. "The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models" [16]. Their github page includes a link to a detection model zoo, which is where SSD Inception V2 was located. A comparison between the SSD model and the YOLO model (which will be explained in 3.3) can be seen below in figure 3.7. The implementation is pretty straightforward. The images were in a separate folder with the annotations (csv files) in another. All that is needed to run came with the API, the changes needed were for instance setting the path to the images and annotations, and changing the config file to match the training you want to do.

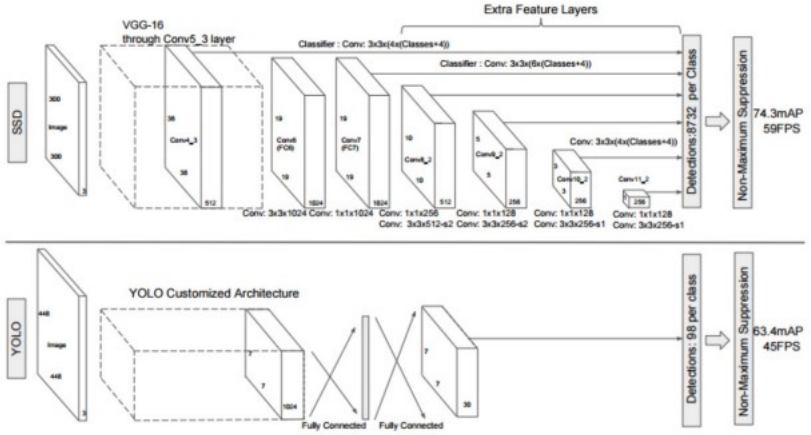


Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300×300 input size significantly outperforms its 448×448 YOLO counterpart in accuracy on VOC2007 `test` while also improving the speed.

Figure 3.7: Comparison between SSD and YOLO [17]

Inception V2 is a way to easier get the correct kernel size for when the object in the image varies in size [18]. It is called V2 as it is a better version of the first Inception. The way Inception helps the model is by having several filters of different sizes operate on the same level. The first Inception module has three different sizes, 1x1, 3x3, and 5x5. It performs convolution on an input with all three filter sizes, with an additional 1x1 convolution *before* 3x3 and 5x5. After this the outputs are concatenated before being sent off to the next module. V2 is just a bit different, with the 5x5 convolution being replaced by two 3x3, as in figure 3.9 to the right. In addition to this, the 3x3 convolutions were split into 1x3 and 3x1 seen in figure 3.8, and then made wider to remove bottleneck, shown in figure 3.10. This results in an architecture looking like 3.11 on the next page. Here figure 5 is equivalent to 3.9, 6 to 3.8, and 7 to 3.10.

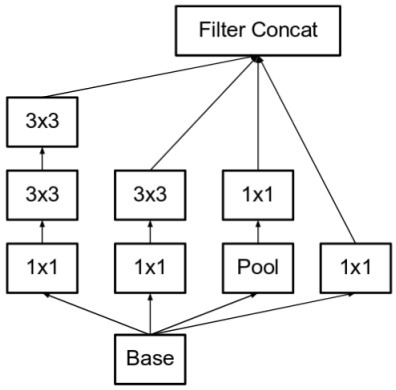


Figure 3.9: Figure 5 in architecture [18]

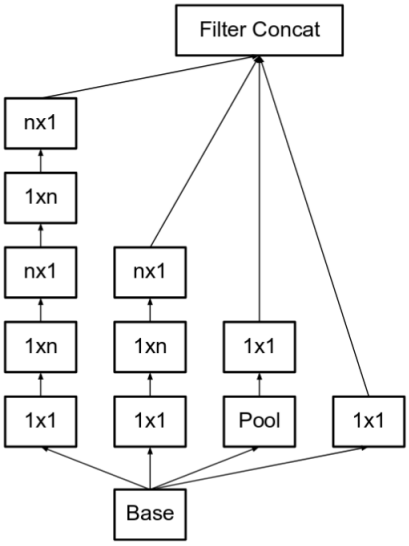


Figure 3.8: Figure 6 in architecture [18]

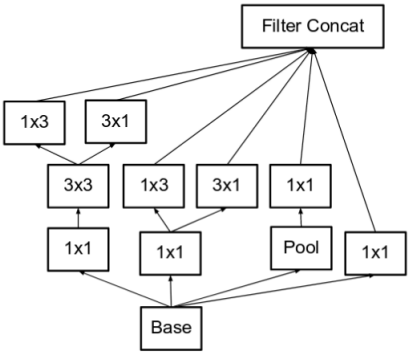


Figure 3.10: Figure 7 in architecture [18]

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Figure 3.11: Inception V2 architecture [18]

3.3 YOLO

You Only Look Once (YOLO) was the second method that was experimented with. The version used in this thesis is the one called Tiny YOLO. This version is much faster than the normal YOLO version, but also less accurate [19]. The implementation of this method was harder than SSD, but only because of different versions based on your operating system. The config file, after downloading it, had to be modified to fit the training planned, just as with SSD. The loss function for this model can be seen further down the page in figure 3.12. To explain it in simple terms, the first line penalizes a bad location of the center of the cells, the second penalizes for errors in height and width of the predicted bounding box. The reason for the square roots in this line is to penalize harder for small errors in small boxes than for small errors in bigger boxes. If there is an object, the third line tries to match the confidence score to that of the IoU between the box and the object.

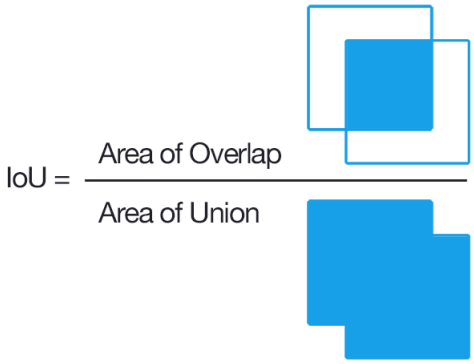


Figure 3.12: Intersection over Union [20]

IoU, or Intersection over Union, is a method to calculate how well a bounding box hit the mark. It is a simple calculation of the area of overlap, the area of the image that is covered by both the ground-truth and the predicted box, divided by the area of union, the area of the image covered by either the ground-truth or the predicted box, or both. This means that if the model tries to be clever and predict on the whole image, it would get a very low score because the area of union would be extremely large. A perfect prediction would give the IoU function a quotient of 1. IoU can be easily understood by looking at figure 3.13 to the left.

Back to the loss function, the fourth line tries to do the opposite of the third line. That is, of there is no object there, it tries to make the confidence score as close to 0 as possible. The fifth and final line is a simple classification loss.

loss function:

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)
\end{aligned}
$$

Figure 3.13: Loss function for YOLO [5]

Chapter 4

Experiments

One big problem when it was time for training was how much time got wasted. The computer with a compatible GPU that was supposed to do the training did not want to run it, and several days were wasted on this. In the end a computer without a compatible GPU was used, which means that all the tests were much slower than what was planned. This is why the tests did not run for as long as they should, and this is probably reflected in the results.

4.1 SSD

The SSD tests were a retraining of a model that was already trained on the COCO dataset. The first couple of runs are not much to talk about as they were quite small, but the third one gave some interesting results. The model trained for 1162 steps, which is not much at all. It would have been preferable to train for ten times as long at least, but that is perhaps something for the future. All the following images are of the full invoice, as the boxes are a bit all over the place. The model predicted a maximum of 4 boxes for these images, though it could have been set as both less and more than that.

In the first image, figure 4.1, we can see that it is hitting pretty close to exactly where the invoice identification number is, with a good 86%. Two of the other boxes are also marking other text fields, but not with a higher certainty than 20% which should not be good enough to be classified. The final box far down on the invoice has only an 11% certainty, so that makes this a good result all in all. The second image on this page, 4.2, is 45% certain that a logo is the invoice identification number. It did find "Samlefaktura" though, which contains the word "faktura", and if that is what it was based on, then that box means that the model is at least connecting "faktura" with the desired field.

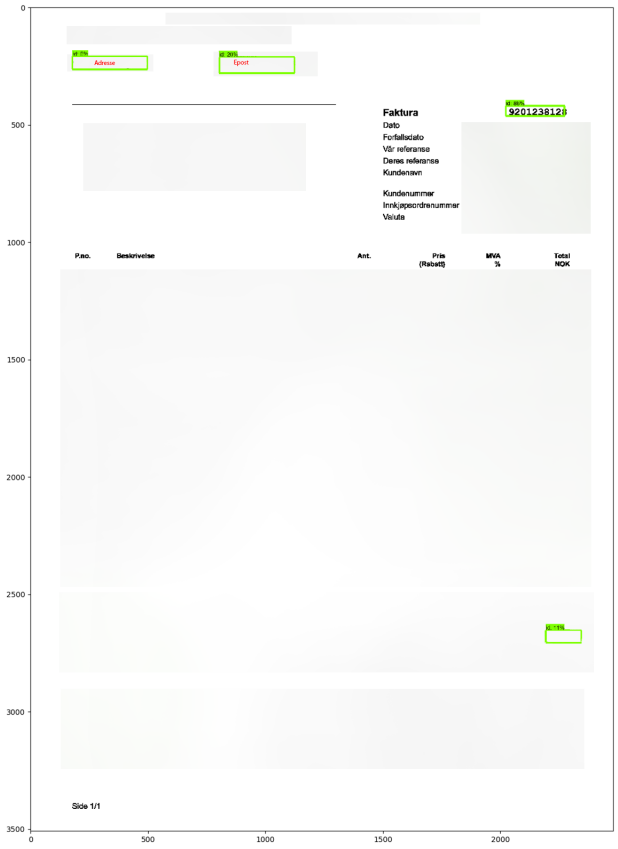


Figure 4.1: A good prediction image

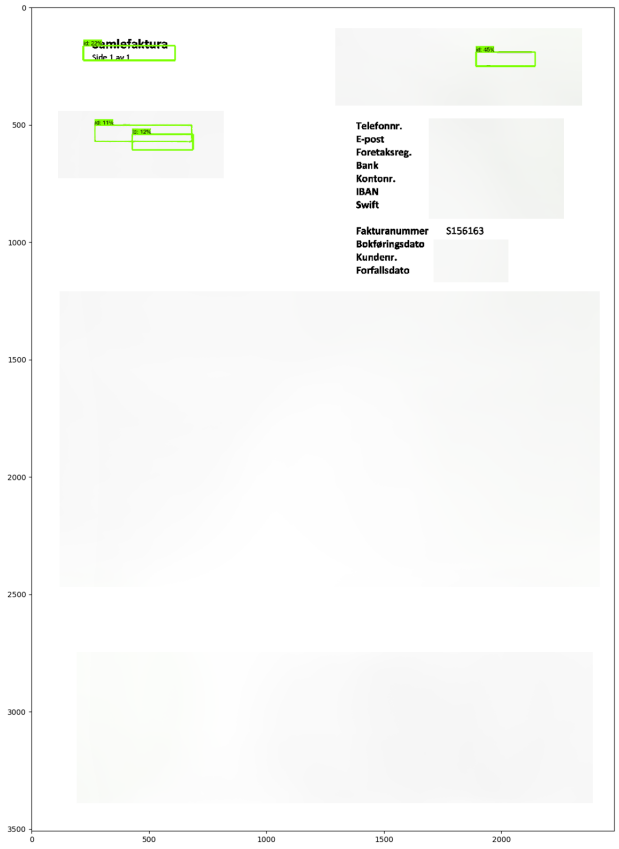


Figure 4.2: Recognizing "faktura"?

Figure 4.3 is not showing good results. It is 61% certain that an address is the identification number, and two of the other three boxes are 30 and 31% certain that a part of a commercial is the number. The last box is no better. It predicts on part of a word, with a low certainty. The other image, figure 4.4, is much more interesting. It predicts a logo and a random word, which is not impressive. It does, however, predict the number to be right above "faktura". The question is if this is random or not. It has a very low certainty of 14%, which could be a hint that it is random, as the previous two boxes also have 14 and 11% respectively. The last predicted box is the most interesting one though. This is the customer identification number, which the box hit quite nicely. Not perfect, but about as good as many of the invoices that actually got their identification number recognized. It is only 21% certain, but it is interesting to see that it looked at a number not too dissimilar to the invoice identification number, and predicted that far down perhaps because it was so similar.

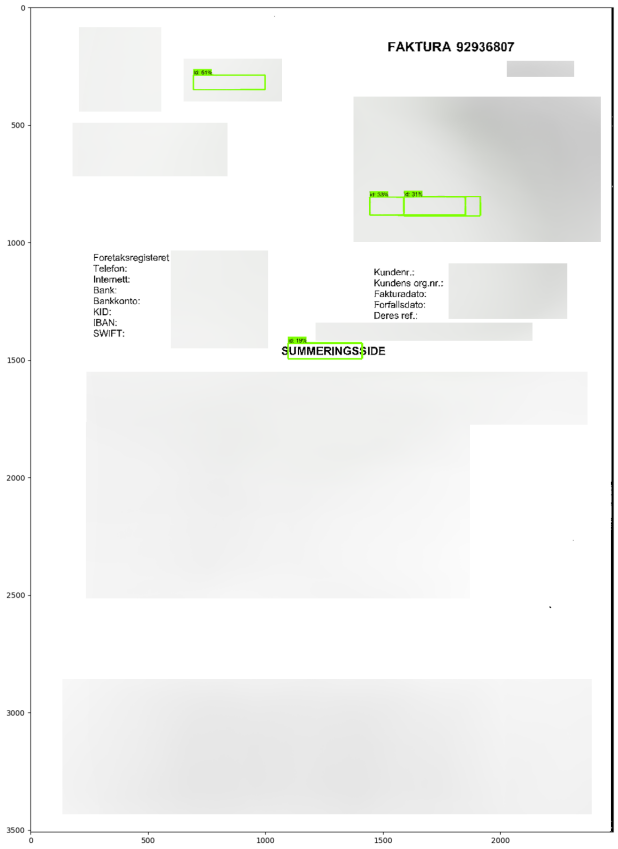


Figure 4.3: A complete miss

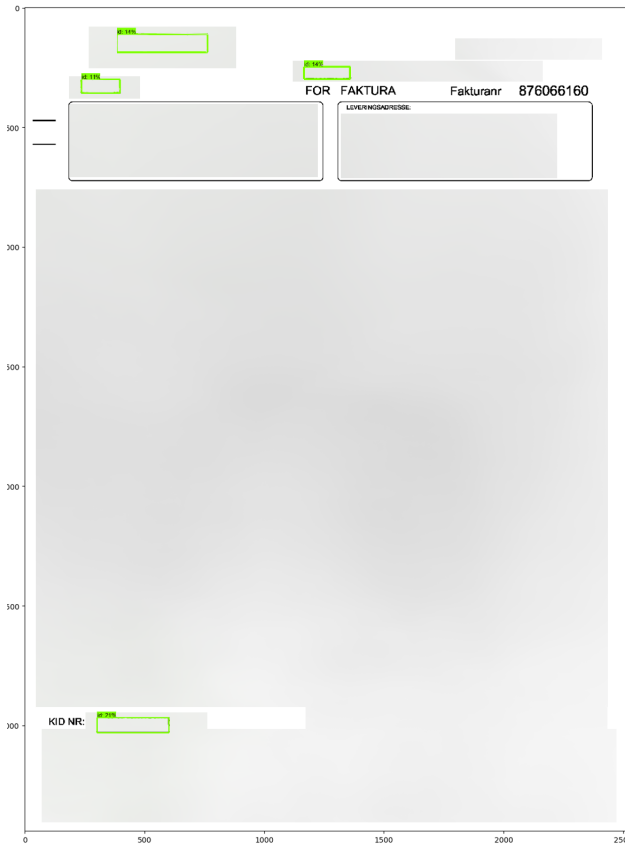


Figure 4.4: Customer identification number

Figure 4.5 is a somewhat good result. Not as good as 4.1, but it still hits the correct number with a 45% certainty. It even predicts the word "faktura" with a 15% certainty, which could make one believe that it understands the connection to this word. The two other boxes further down are a telephone number and an address, which are not very interesting. The next figure, 4.6, is worse again. It finds "faktura" with 40%, but not the number beside it as 4.5 did. It is also 30% certain that a logo is the number it wants to find, which does not look good. It also finds part of a word further down, and part of a bank account number. Again, it finds a long number that can remind of the invoice identification number. It is only 11% certain of that placement, but it is interesting that it seems to sometimes recognize long numbers as what it is looking at.

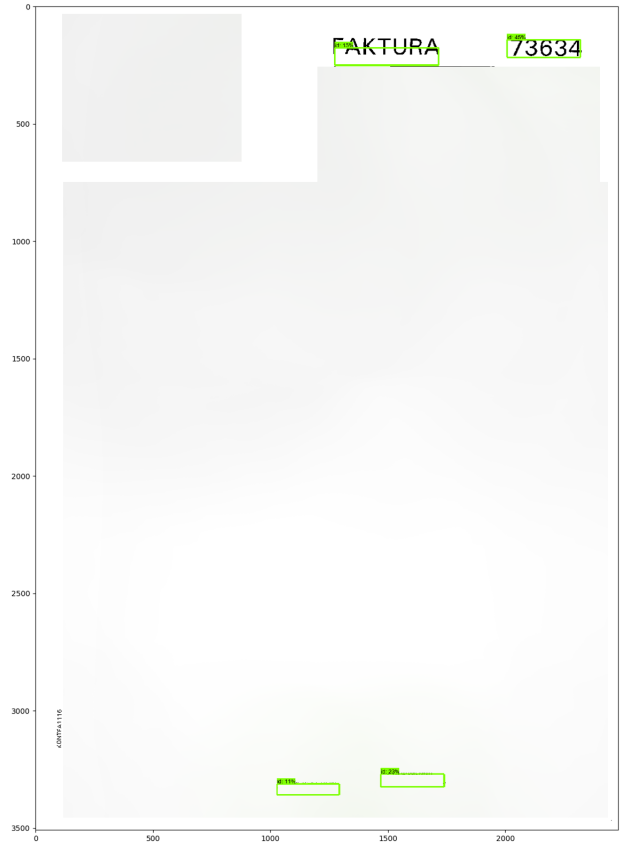


Figure 4.5: Finds "faktura" and the invoice identification number

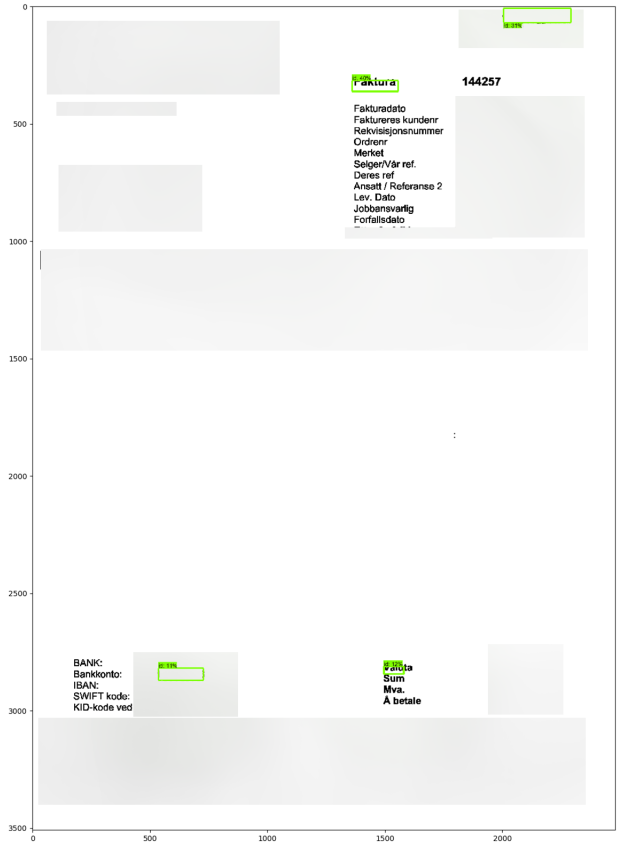


Figure 4.6: Finds "faktura", but not the number beside it

This final image that is figure 4.7 is again showing that it recognizes "faktura". It is much surer that the logo to the right or especially the name to the left under "faktura" is the correct location, with only being 26% certain of "faktura", 11 and 44% certain of the logo, and 67% sure of it being the name.

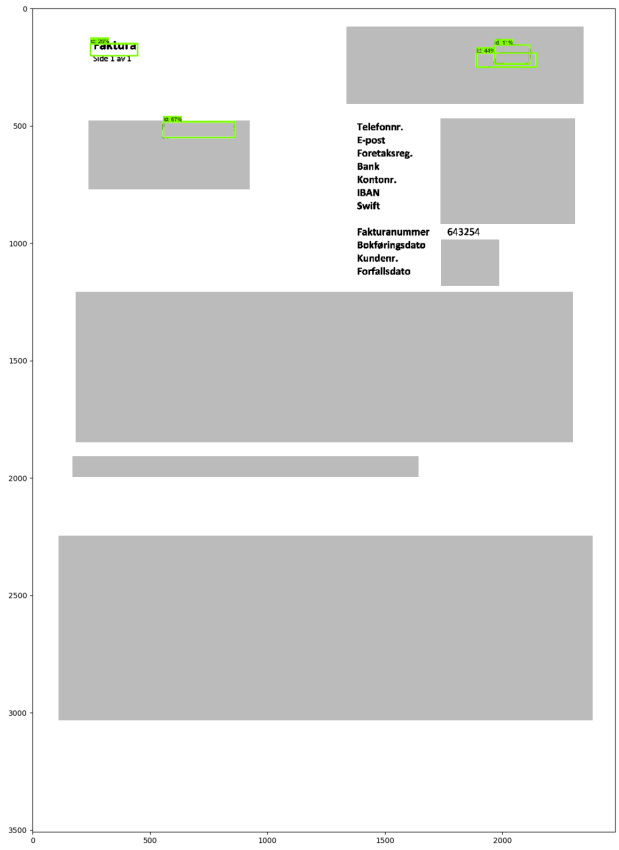


Figure 4.7: Another "faktura"

All in all it shows that this should be tested further. The fact that it repeatedly finds "faktura" can be taken as a sign that it is on its way to understanding where to look, in addition to the times where the model finds those long numbers. It does not always get even close to the correct location, but that could come with more training.

4.2 YOLO

4.2.1 Training with the Invoice Dataset

There were three experiments with this method. First it was a basic retraining of an existing model on 90% (2700) of the data for exactly 1000 steps, but this did not produce the desired outcome. A few examples of the test invoices can be seen in figures 4.8 to 4.13 below and on the next page. All the figures together show that the model does not understand what it is looking at. The predicted box is always in one of two locations, with some variation on the dimensions. In figure 4.10 we can see this very clearly, as the box is around nothing. Figure 4.8, however, shows that in addition to the "normal" box location, it also predicts close to the actual invoice identification number. This could be a sign that the model tries to actually learn what it is supposed to learn, it just has not been given enough time to unlearn the most likely position for the number to be. Some of the prediction boxes are close to the number, but even those miss slightly, as we can see from figure 4.12 and 4.13. Figure 4.12 does get a box that covers the whole number in addition to the box that only gets the top of it, so that is far from the worst image. The invoices like figures 4.10, 4.12, and 4.13 are a significant part of the training data, so it makes sense that this is what it is overfitting on, but even those invoices are a little bit off. 4.9 and 4.11 are examples where the prediction actually hits text (in the form of numbers), but unfortunately not the correct field.

Figure 4.8: Two boxes, with one being somewhat close

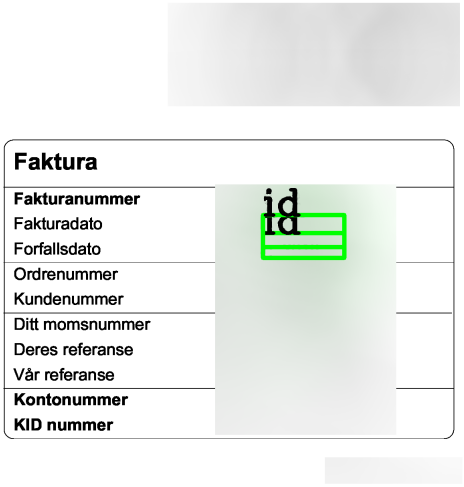


Figure 4.9: Missing the invoice identification number

Figure 4.10: Predicting a blank space

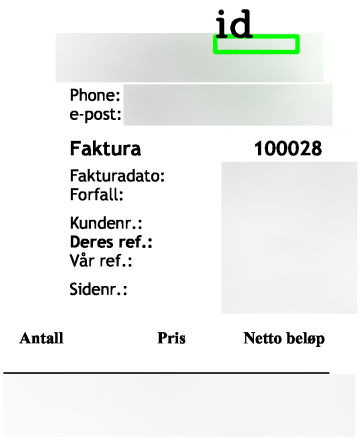


Figure 4.11: Missing the invoice identification number

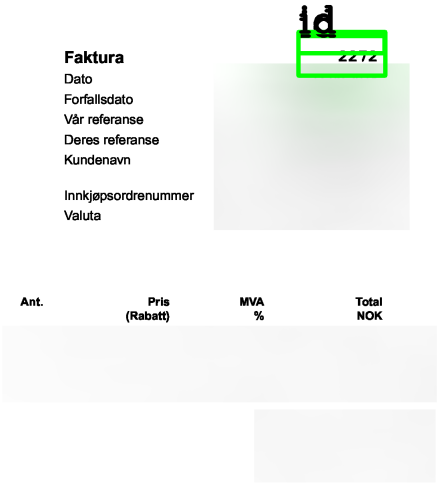


Figure 4.12: Correct prediction in addition to missing almost completely

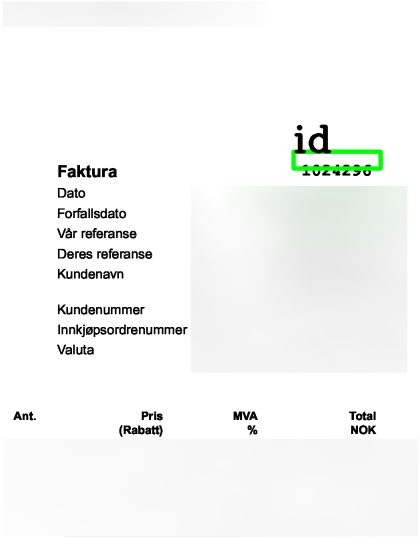


Figure 4.13: Only hitting part of the numbers

The threshold for these boxes was set as 0.4, which resulted in 199 of 300 invoices getting one or more prediction boxes. This means that there are 101 images where the model was not sure enough of any box, which is disappointing. If the threshold is set lower then more images would get a prediction on the invoice identification number, but it would not be any better boxes. If anything the images would just get filled with more boxes in the same corner.

4.2.2 With Rotation to Combat Overfitting

As the results from the previous experiment were not nearly good enough, the training data was also rotated 90, 180, and 270 degrees to try to combat overfitting and give more data to train on. This did not seem to make any significant difference, though the training took much longer because of the quadrupling of the training data. When looking through the images, the box as good as did not move other than changing size. This is probably because of training for a very short time, with only 620 steps. Mostly it predicts the topmost text on the right side, for instance in figure 4.15, though there are some exceptions as we can see from figure 4.14. Although this test was run for only half the amount of steps as the first test, the results were not much worse, considering that most of the test data fit that one place.

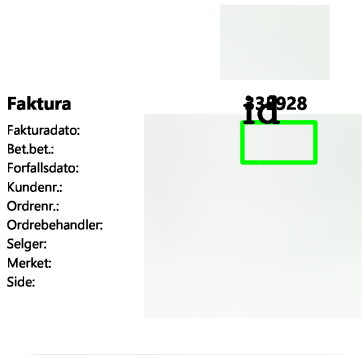


Figure 4.14: Predicting too low

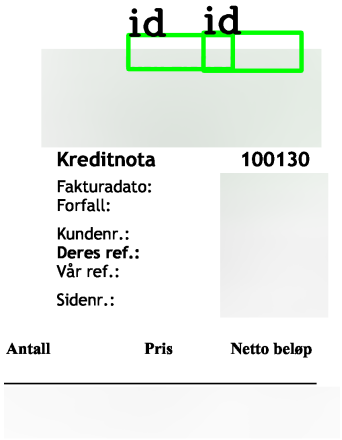


Figure 4.15: Too high, the first text from the top

It does predict quite nicely on most of the invoices that did get a prediction, but that is only because of the invoice identification number conveniently being in the correct place. This is the type of invoice the training data has the most of, which is most likely why it is predicting the way it is. The rotation of images apparently did not help with the overfitting. The likely reason is that the model recognizes the rotation based on where the list of fields are located, as these are quite recognizable, and then predicts the correct corner based on the rotation. The invoice format that was the easiest to test on was the same as the last test, which makes sense as it is the same training data just with the rotated images as well.

It is not very sure of itself yet, with only 138 of the 300 invoices getting boxes with a threshold of 0.5. When the threshold was set to 0.4, 172 images get their prediction, but the quality of the additional images was way worse. With 0.5 it predicted primarily on two versions of the invoices, while with 0.4 a couple other formats showed up. Two examples of this can be seen in figures 4.16 and 4.17 to the right. As we can see, these are not good examples of a job well done.

The loss of this experiment went down to between 2 and 4 before being stopped. This is about the same as the first experiment. It had been stable for the last hundred or so steps, though it was still not as good as hoped. The next experiment tried to give it time to lower itself further.

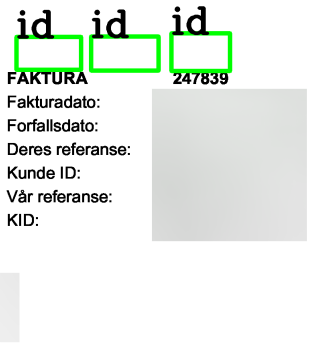


Figure 4.16: An example of the new invoice format when threshold is set as 0.4

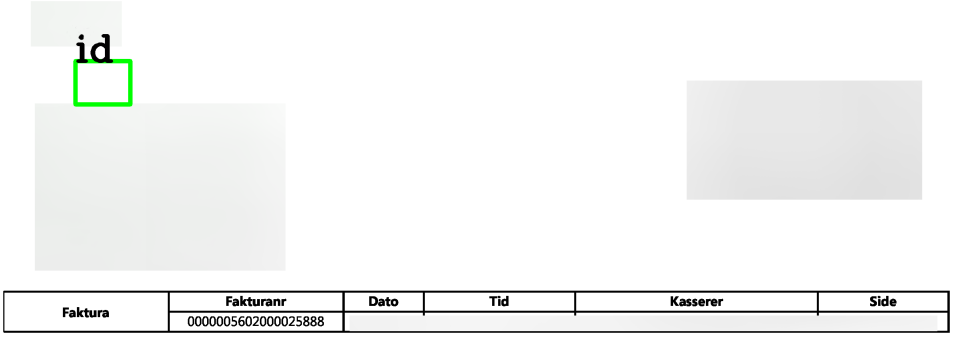


Figure 4.17: Missing on an invoice with threshold 0.4

4.2.3 Training for Longer With Rotation

The last attempt was actually just a continuation of the previous one. Instead of running for 620 steps, it was allowed to run for 3844 steps. This did give a better result, but only somewhat. Instead of predicting on the same spot over and over, it now *only* predicts on the easier invoices. This means that it is getting better on the popular invoice formats, but not on the others. This result looks better because now it actually hits some of the correct locations, but it is still quite bad since it predicts on such a small number. This just means that it is learning that specific invoice even better than before, which would be good if that was the only type of invoice available, but it does not work at all when testing it on other formats.

Six examples of the results from this experiment can be seen below and on the next page in figures 4.18 to 4.23. When setting the threshold to 0.4, 98 of the images got one box. Even setting the threshold to 0.2 did not give a single invoice more than one box. If the threshold is set at 0.6, only 57 of the 300 test images get any prediction box, and all 57 of them are of the same format. This is the only layout this model can predict somewhat close to correct on. All the examples from this experiment include the confidence underneath.

The first two figures, 4.18 and 4.19, show two examples of the kind of invoice that gets a close prediction. As we can see, the one on the right is not completely around all the numbers, but it is still fairly close.

Figure 4.18: Confidence 0.67

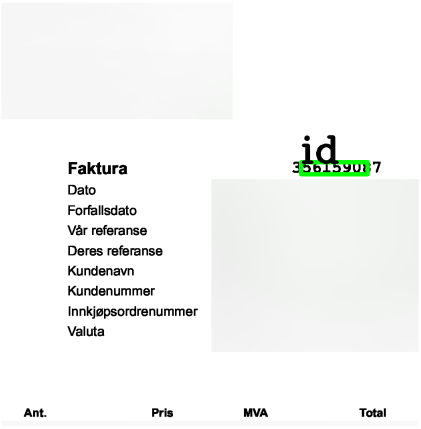


Figure 4.19: Confidence 0.66

Figure 4.20 also hits pretty well, with only a small amount of air around the number. It also has one of the highest confidence scores of all the images at 82%, with the highest being 85%. Figure 4.22 is the same layout as all the others that the model managed to predict on, but somehow its confidence is surprisingly low at only 57%. Figures 4.21 and 4.23 are examples of how the model is not predicting correctly when it gets surprised by an unfamiliar layout.

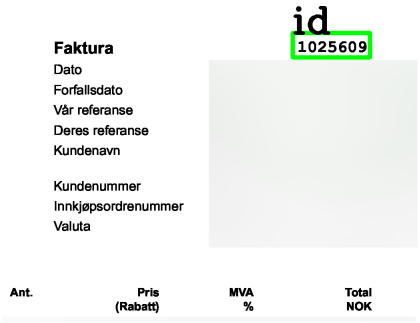


Figure 4.20: Confidence 0.82

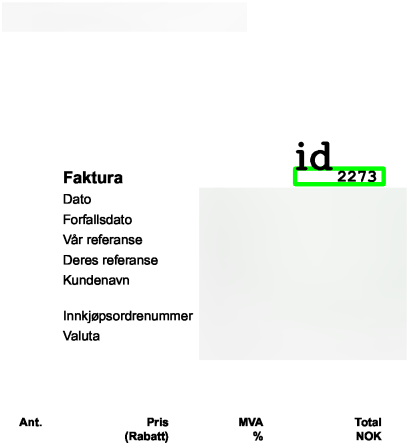


Figure 4.22: Confidence 0.57

Figure 4.21: Confidence 0.42

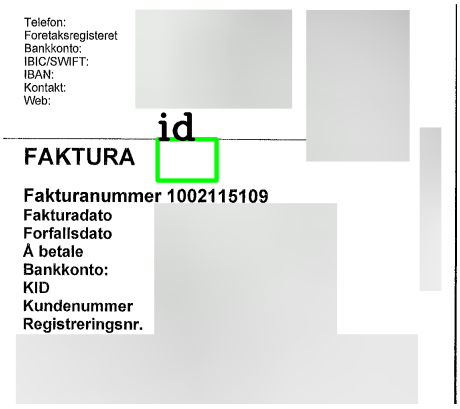


Figure 4.23: Confidence 0.44

The loss for this test did not get much lower than the previous ones. For the first 1000 steps since continuing from step 620 it hovered right on the edge between 1 and 2, and stayed mostly between 1 and 3 for the rest of the training. It could be a plateau that is possible to get past, but it could also be that it has specialized too much on the few formats that come up often and is struggling to stop. The plateau could be the end, with the model just getting better and better on those few formats. It is also possible that it would start overfitting on the other formats in its training data too, but it would still go against the goal of this thesis to have a generalized model that can find the invoice identification number on a brand new format, which this model absolutely cannot.

Chapter 5

Conclusion

This thesis tests if it is possible to predict an object (invoice identification number) in a document (invoice) without the use of OCR, and where the layout is unknown. This did not happen, but could possibly have an easy solution. The two methods used for this thesis show different results, but both fail. Both could be because of not enough training.

The tests done with SSD shows that with an easy invoice the identification number can be recognized. It looks like it understands some of the basics of looking for "Faktura" ("Invoice") even with the invoices of the format type it does not confidently predict on yet, which can be seen as a good sign. Hopefully this is a model that can be trained further to get better results.

As mentioned, with SSD many of the same type of invoice got a pretty good prediction. This should mean that this was the kind of invoice that had the most data to train on, and if this is the case then other invoices should also be possible to get right with enough data. The downside to this is that it would mean the training was not generalized enough, and that a new invoice with a completely new layout would not get a good prediction. This would go against the desired outcome of this thesis. As there is hope even with the other data, this should be explored further before giving it up as a lost cause. Making the training data more diverse could also help.

The tests done with YOLO give results mostly showing boxes at the first black spots in the top right corner, whether they be the identification number, which would be right, a logo, a name, or another field with words or numbers. There was an attempt to correct this by rotating the dataset 90, 180, and 270 degrees, but this showed little improvement. This is probably overfitting towards the most popular invoice layout, which could possibly be solved by more varied data, or it could mean that YOLO is not the way to go to solve this problem. The third test got even more specialized towards the easier invoice format, further hinting towards this not being possible with YOLO, at least not with the training data being as is.

These tests did not show a lot of promise, and it is possible that they never will. The times where the prediction is nothing of interest, for instance a logo, it tells us that it does not understand what it is looking at. The times where it predicts other kinds of information, however, is where we could find hope. If the tests were to include more than only the identification number the model could be forced to be more selective in what information is predicted as being the different classes. Based on the experiments in this thesis YOLO does not seem like a good match for this type of object detection, but SSD Inception V2 could be further tested as this did show some promise.

This thesis indicates the start of getting information from a document without the use of OCR, but the results did not live up to our expectations. The results from the tests do not show certainty or precision, but they tell us that it could be possible eventually.

Chapter 6

Future work

The results in this thesis are disappointing. For it to be possible to work with, at least one of the models should be somewhat reliable. For now, the focus should be on improving the accuracy and generalization of the two models.

There are many possibilities to improve the results from both SSD and YOLO. The simplest solution would be to spend more time on training. This might not be enough though, as both methods show some signs of overfitting on the most common invoices and not generalizing it to work on more unusual invoices. More varied training data should be tested, even if it is just cutting them in half horizontally and switching the halves. Some variation in the parameters should also be tested. Seeing as the tests done in this thesis were time consuming, there was not enough time to experiment with these.

Another solution to getting the models to generalize could be to add other classes. Using YOLO, on quite a few of the test data the most confident box would be the topmost field of the field list. This means that if the invoice identification number is in a different location or even if it is a different size than the other fields, then the topmost field would be wrongly tagged as the identification class. If more classes are being identified, then the model would get the possibility to ignore the fields that are already classified as something else, and would have to look elsewhere for the identification number. For this to work, the actual field would have to get a higher certainty from the model so as to "win" over the identification number, in addition to the model having enough time to learn that there is never an overlap between the fields.

References

[1] http://cs231n.github.io. " CS231n: Convolutional Neural Networks for Visual Recognition.". [Online], Available: http://cs231n.github.io/convolutional-networks, Visited: 20.05-2019.

[2] Michael Nielsen. "Using neural nets to recognize handwritten digits". [Online], Available: http://neuralnetworksanddeeplearning.com/chap1.html, Visited: 20.05-2019.

[3] Jonathan Hui. "SSD object detection: Single Shot MultiBox Detector for real-time processing". [Online], Available: https://medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06, Visited: 20.05-2019.

[4] Mauricio Menegaz. "Understanding YOLO". [Online], Available: https://hackernoon.com/understanding-yolo-f5a74bbc7967, Visited: 20.05-2019.

[5] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". [Online], Available: https://arxiv.org/pdf/1506.02640.pdf, Visited: 23.05-2019.

[6] Mike James. "You Only Look Once - Fast Object Detection". [Online], Available: https://www.i-programmer.info/news/105-artificial-intelligence/11061-you-only-look-once-fast-object-detection.html, Visited: 20.05-2019.

[7] "Object Detection on Coco". [Online], Available: https://paperswithcode.com/sota/object-detection-on-coco, Visited: 20.05-2019.

[8] Yanghao Li et al. "Scale-Aware Trident Networks for Object Detection". [Online], Available: https://arxiv.org/pdf/1901.01892v1.pdf, Visited: 20.05-2019.

[9] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". [Online], Available: https://arxiv.org/pdf/1506.01497.pdf, Visited: 23.05-2019.

[10] Yingyi Sun et al. "Template Matching-Based Method for Intelligent Invoice Information Identification". [Online], Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8653814, Visited: 24.05-2019.

[11] Ankit Sachan. ”Zero to Hero: Guide to Object Detection using Deep Learning: Faster R-CNN,YOLO,SSD”. [Online], Available: https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/, Visited: 23.05-2019.

[12] ”Invoice Capture Software – Is Automated Invoice Scanning a Viable Solution?”. [Online], Available: https://docparser.com/blog/invoice-scanning/, Visited: 24.05-2019.

[13] Xavier Holt and Andrew Chisholm. ”Extracting structured data from invoices”. [Online], Available: https://www.aclweb.org/anthology/U18-1006, Visited: 24.05-2019.

[14] ”Sparebank 1 Invoice Scan (in Norwegian)”. [Online], Available: https://nyhetssenter.sparebank1.no/bv/2018/06/19/ny-superenkel-regningsbetaling-i-mobilbanken/, Visited: 24.05-2019.

[15] ”DNB Invoice Scan (in Norwegian)”. [Online], Available: https://www.dnb.no/mobilbank, Visited: 24.05-2019.

[16] Jonathan Huang et al. ”Tensorflow Object Detection API”. [Online], Available: https://github.com/tensorflow/models/tree/master/research/object_detection, Visited: 24.05-2019.

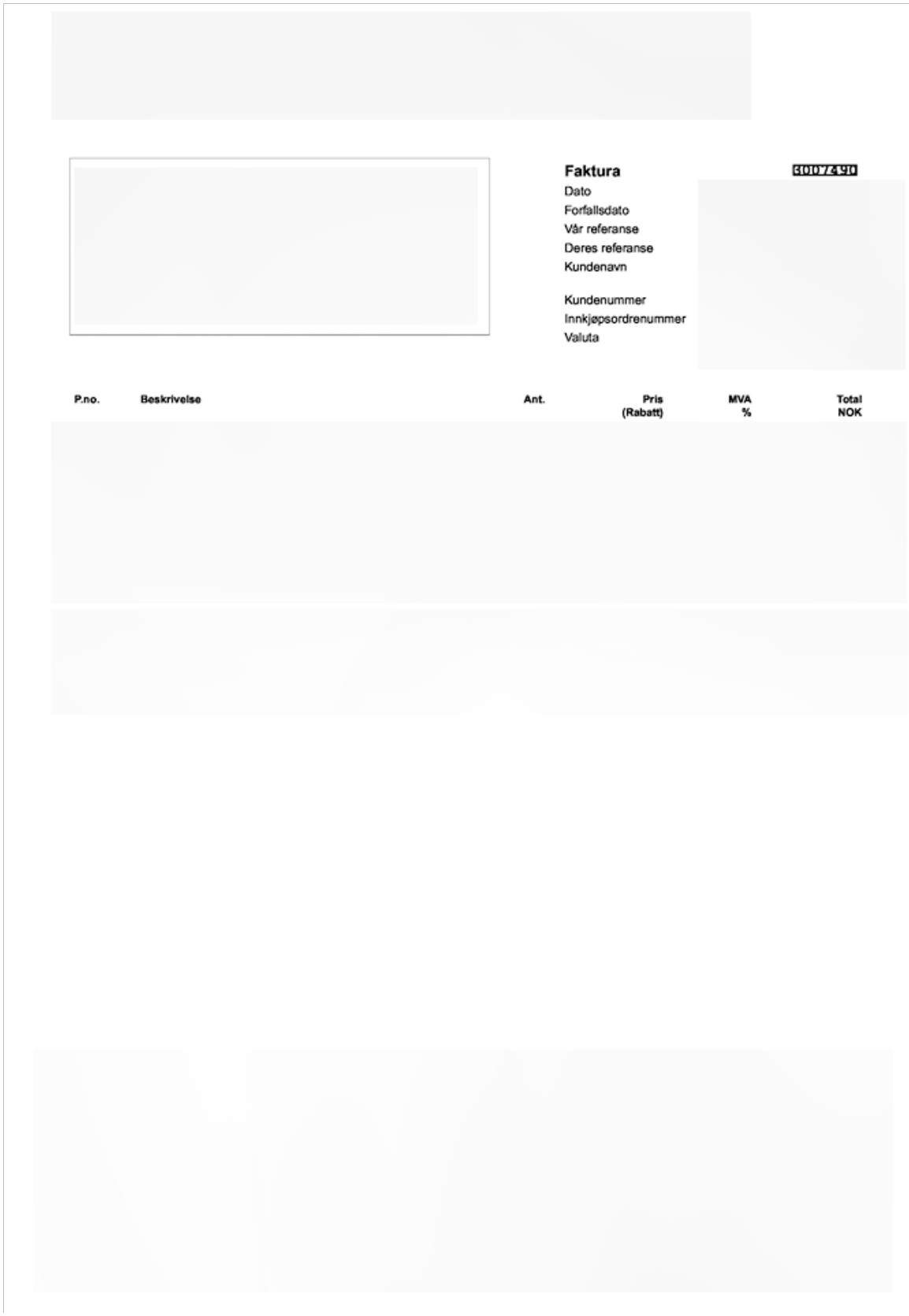
[17] Wei Liu et al. ”SSD: Single Shot MultiBox Detector”. [Online], Available: https://arxiv.org/pdf/1512.02325v5.pdf, Visited: 24.05-2019.

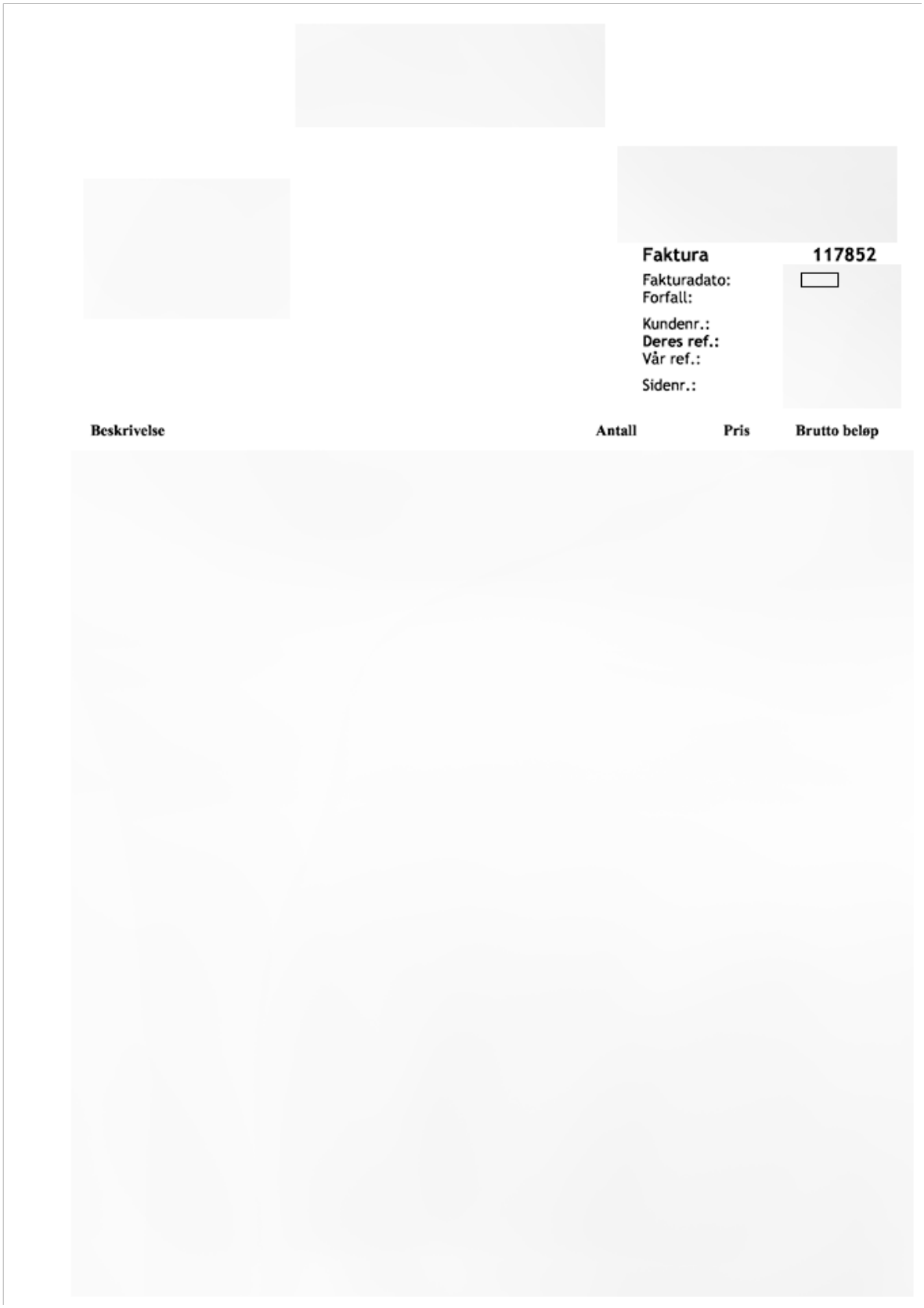
[18] Bharath Raj. ”A Simple Guide to the Versions of the Inception Network”. [Online], Available: https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202, Visited: 24.05-2019.

[19] ”This is Darknet implementation of YOLO - Object Detection”. [Online], Available: https://github.com/shivajid/Yolo, Visited: 24.05-2019.

[20] Adrian Rosebrock. ”Intersection over Union (IoU) for object detection”. [Online], Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/?fbclid=IwAR12FdzJKieoNTWTxIMUbHQtVpiaEUr1fLbBjPAfyOEGPtYsZ5ag8ZKQnIM, Visited: 24.05-2019.

Appendix A


Faktura
Dato
Forfallsdato
Vår referanse
Deres referanse
Kundenavn
Kundenummer
Innkjøpsordrenummer
Valuta
P.no.
Beskrivelse
Ant.
Pris
(Rabatt)
MVA
%
Total
NOK


Faktura
117852
Fakturadato:
Forfall:
Kundenr.:
Deres ref.:
Vår ref.:
Sidenr.:
Beskrivelse
Antall
Pris
Brutto beløp

Periode
Dato
Kundenr.
Organisasjonsnr.

Eksp.dato	Eksp.nr	Løpenr.	Deklarant	Statistisk verdi	Avgift

Invoice ID number

Faktura 1545824

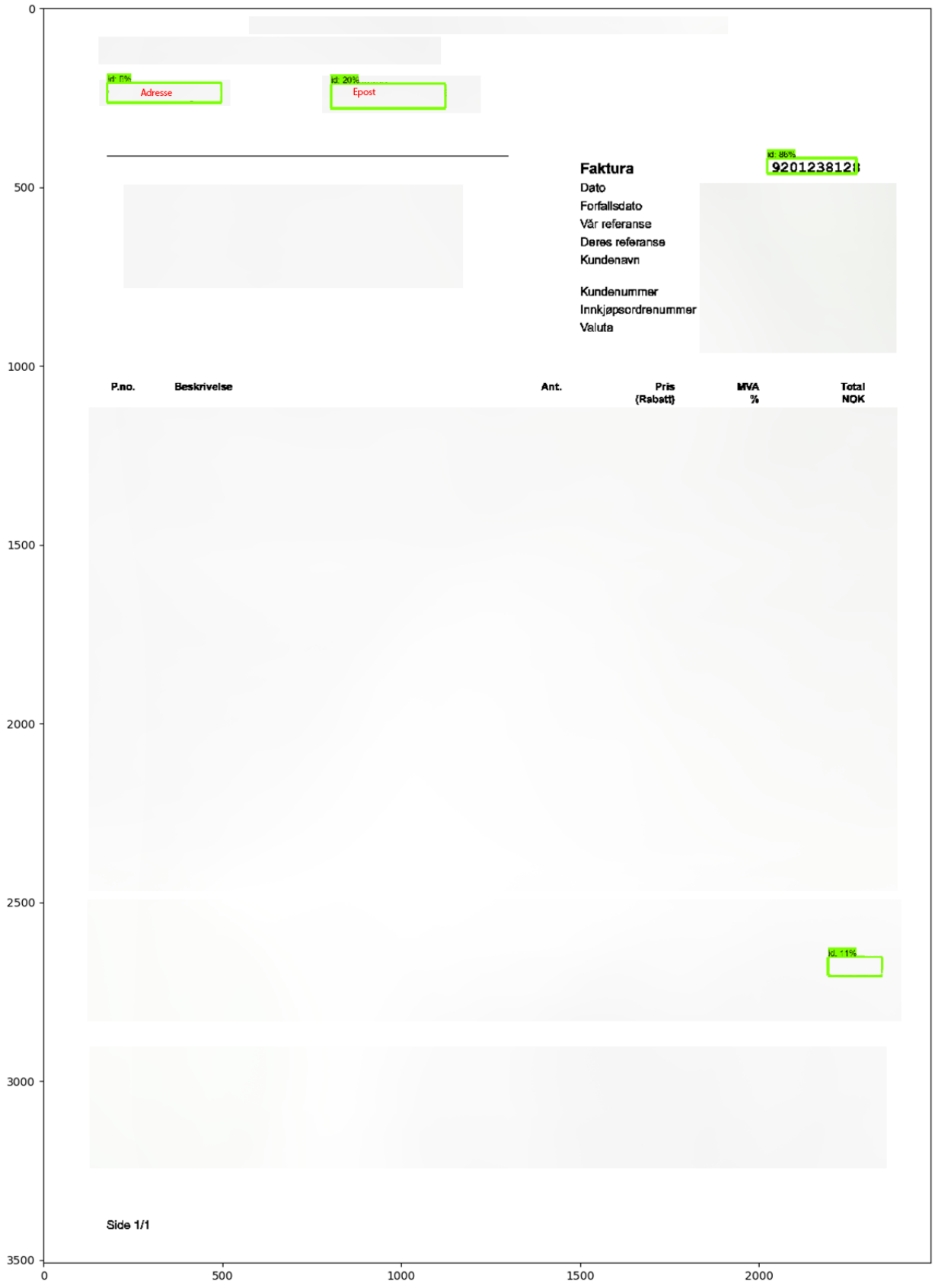
Dato
Forfallsdato
Vår referanse
Deres referanse
Kundenavn

Kundenummer
Valuta

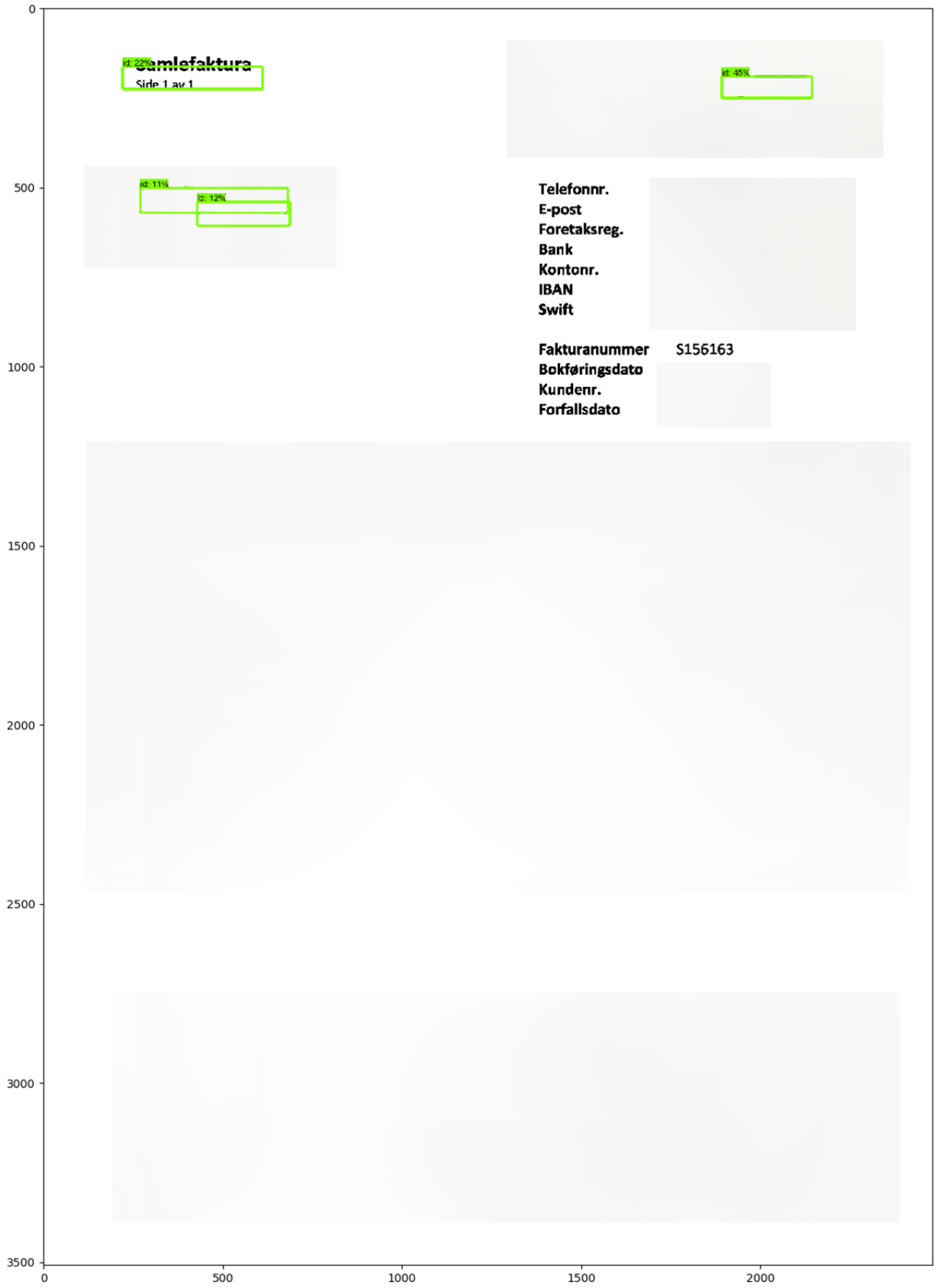
P.no.	Beskrivelse	Ant.	Pris (Rabatt)	MVA %	Total NOK

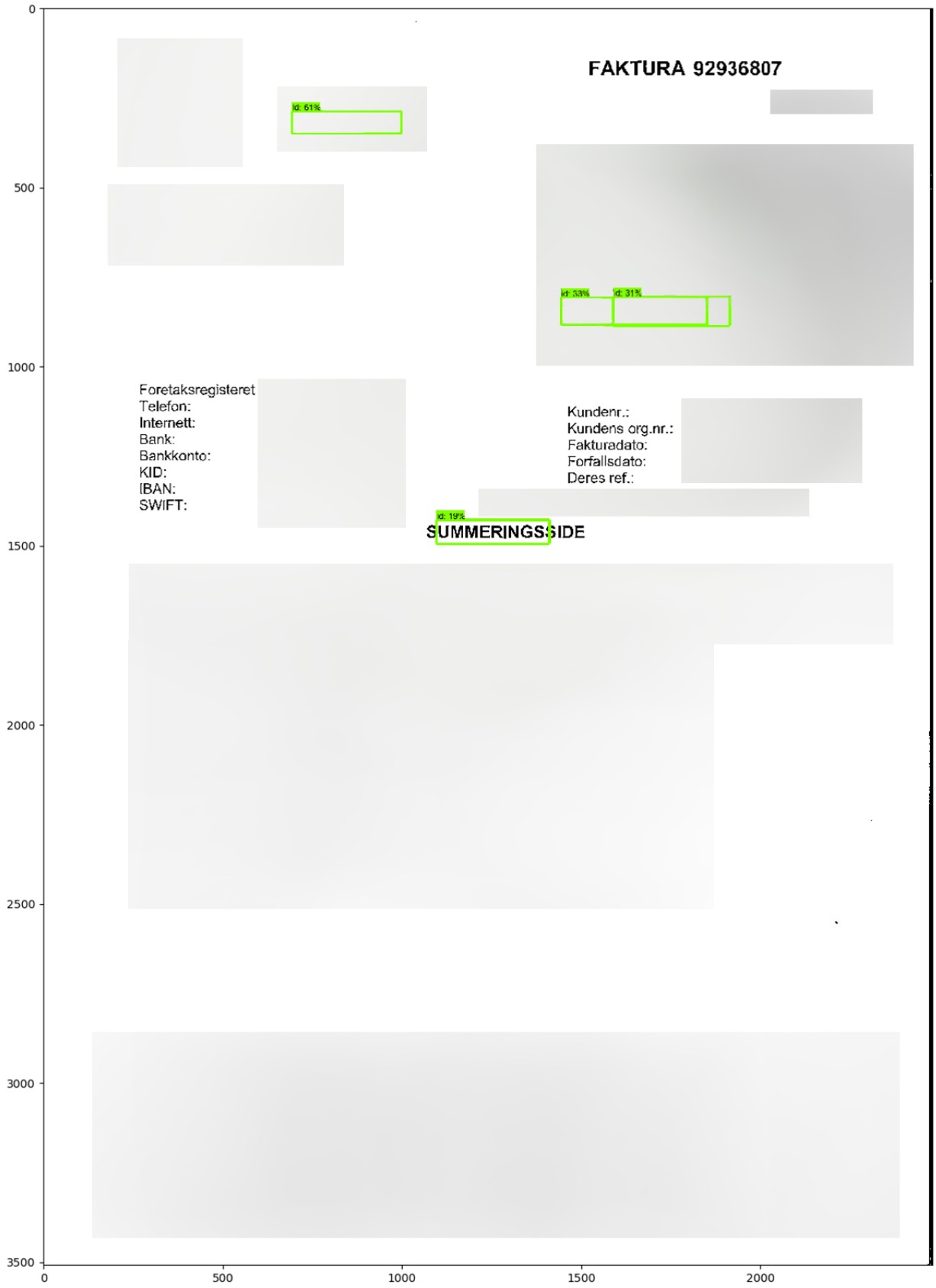
Faktura 18638

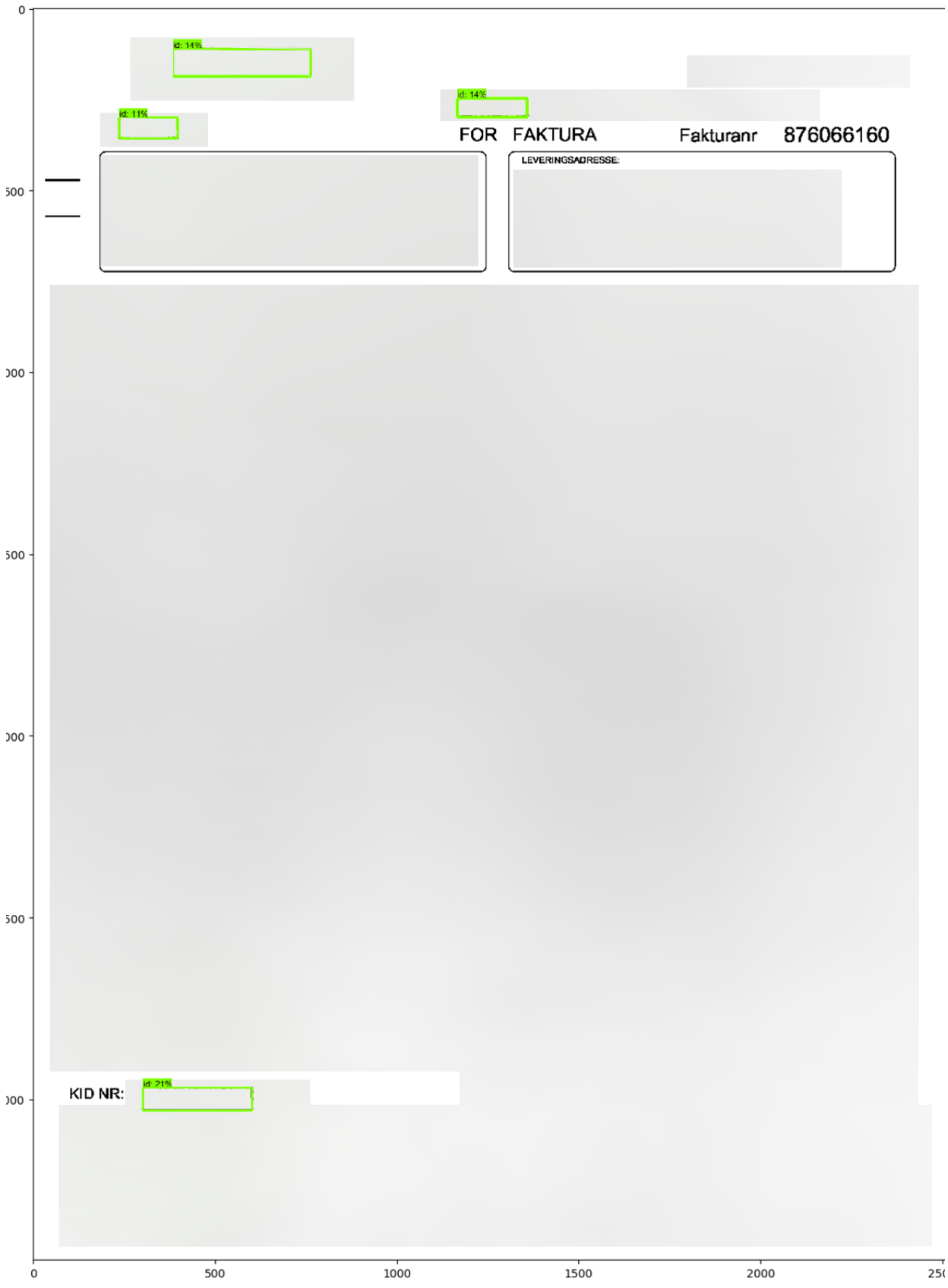
Fakturadato:
Leveringsdato:
Forfallsdato:
Vår referanse:
Deres referanse:
Bilagsnr:
Bankgiro:
IBAN
SWIFT
Ref. fakturanr
Rekvisisjon


Adresse
Epost
Faktura
Dato
Forfallsdato
Vår referanse
Deres referanse
Kundenavn
Kundenummer
Innkjøpsordrenummer
Valuta
P.no.
Beskrivelse
Ant.
Pris
(Rabatt)
MVA
%
Total
NOK
Side 1/1

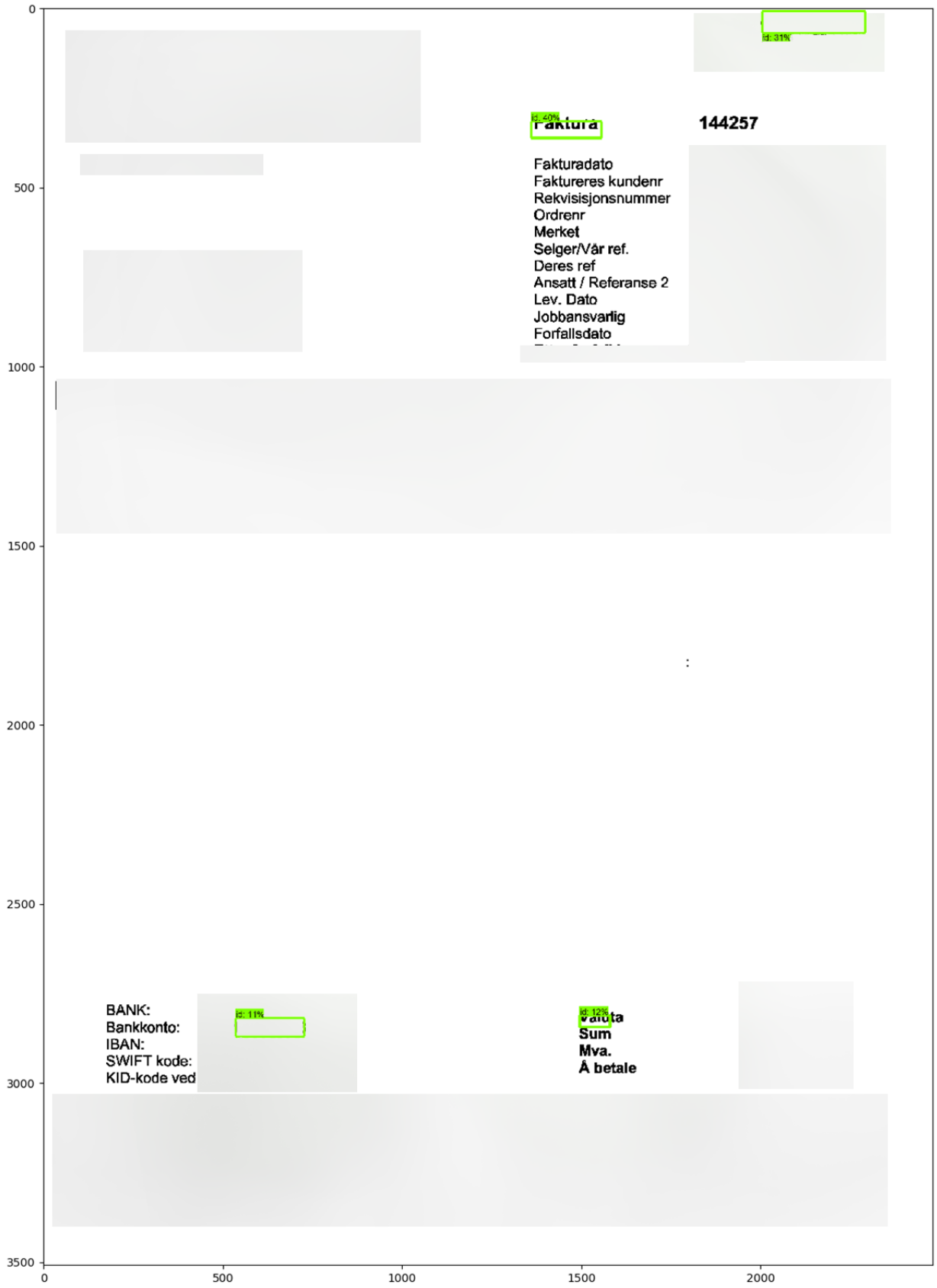
4.1

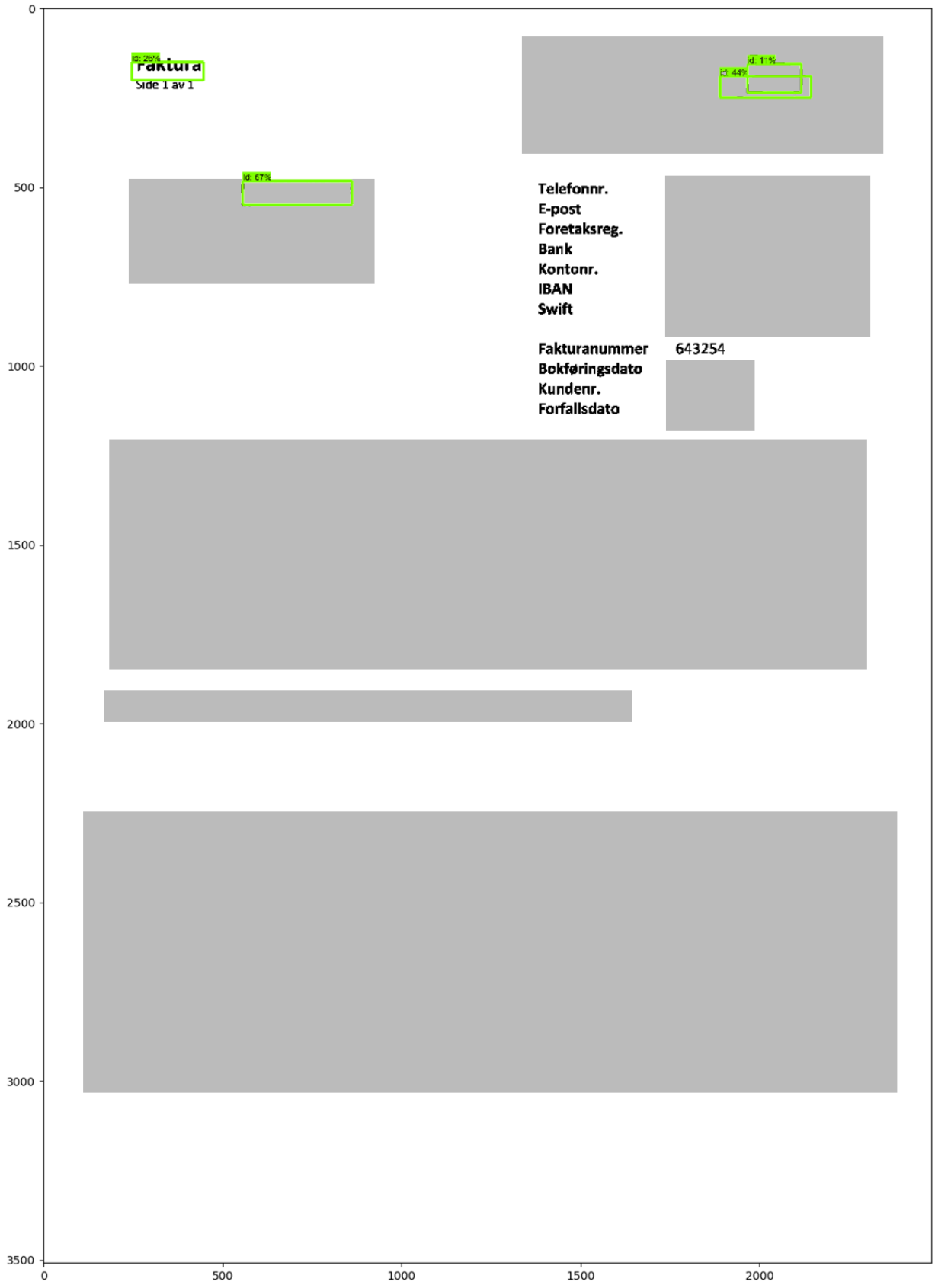

Samlefaktura
Side 1 av 1
Telefonnr.
E-post
Foretaksreg.
Bank
Kontonr.
IBAN
Swift
Fakturanummer S156163
Bokføringsdato
Kundenr.
Forfallsdato
0
500
1000
1500
2000
2500
3000
3500
0
500
1000
1500
2000


FAKTURA 92936807
id: 61%
id: 33%
id: 31%
Foretaksregisteret
Telefon:
Internett:
Bank:
Bankkonto:
KID:
IBAN:
SWIFT:
Kundenr.:
Kundens org.nr.:
Fakturadato:
Forfallsdato:
Deres ref.:
id: 19%
SUMMERINGSSIDE
0
500
1000
1500
2000
2500
3000
3500
0
500
1000
1500
2000


id: 14%
id: 14%
id: 11%
FOR FAKTURA
Fakturanr 876066160
LEVERINGSADRESSE:
KID NR:
id: 21%
0
500
1000
1500
2000

4.5


Faktura
144257
Fakturadato
Faktureres kundenr
Rekvisisjonsnummer
Ordrenr
Merket
Selger/Vår ref.
Deres ref
Ansatt / Referanse 2
Lev. Dato
Jobbansvarlig
Forfallsdato
BANK:
Bankkonto:
IBAN:
SWIFT kode:
KID-kode ved
Valuta
Sum
Mva.
Å betale
0
500
1000
1500
2000
2500
3000
3500


Faktura
Side 1 av 1
Telefonnr.
E-post
Foretaksreg.
Bank
Kontonr.
IBAN
Swift
Fakturanummer 643254
Bokføringsdato
Kundenr.
Forfallsdato
0
500
1000
1500
2000
2500
3000
3500

id

Phone:
e-post:

id

Faktura 100118

Fakturadato:
Forfall:

Kundenr.:
Deres ref.:
Vår ref.:

Sidenr.:

Beskrivelse	Antall	Pris	Netto beløp

	Fritt	12% mva	15% mva	25% mva	Avrunding
Netto					
Mva					
Brutto					

Å betale:
Bankkonto:
KID:
Orgnr.:

Faktura

Fakturanummer	id
Fakturadato	
Forfallsdato	
Ordrenummer	
Kundenummer	
Ditt momsnummer	
Deres referanse	
Vår referanse	
Kontonummer	
KID nummer	

Vilkår:

Item	Vare	Pris eks mva	Ant	Rabatt	Total eks mva	Mva

Pristype	Sats	Grunnlag	Total

Navn:
Momsnummer:
Kontoradresse:
Postboks:

id

Faktura **88260**

Kundenr.
Betalingsbet.
Deres referanse
Fakturadato
Forfallsdato
Kontaktperson
Rekvisisjonsnr.
Selger

Nr.	Beskrivelse	Salgspris	Beløp

id

Phone:
e-post:

Faktura **100028**

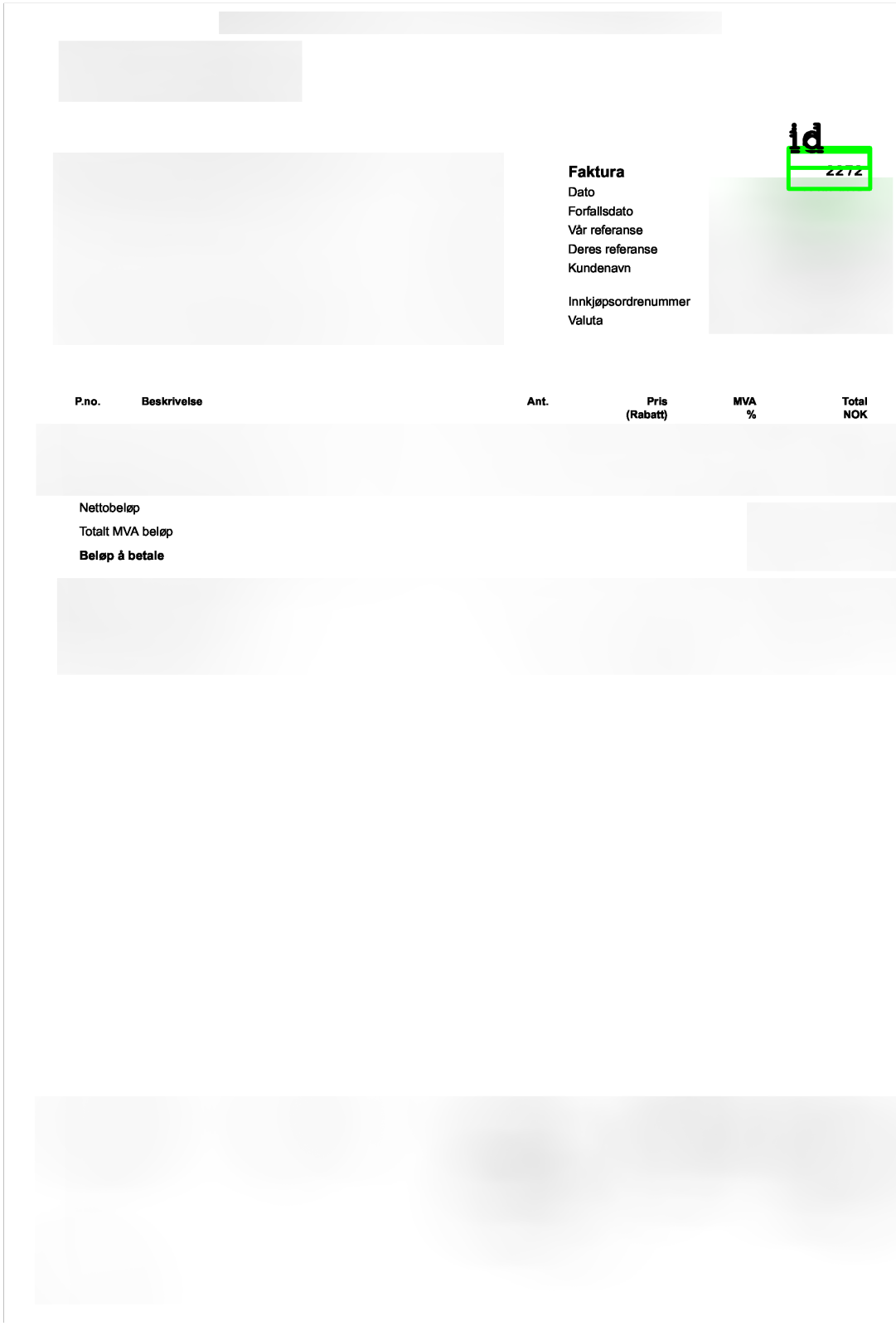
Fakturadato:
Forfall:

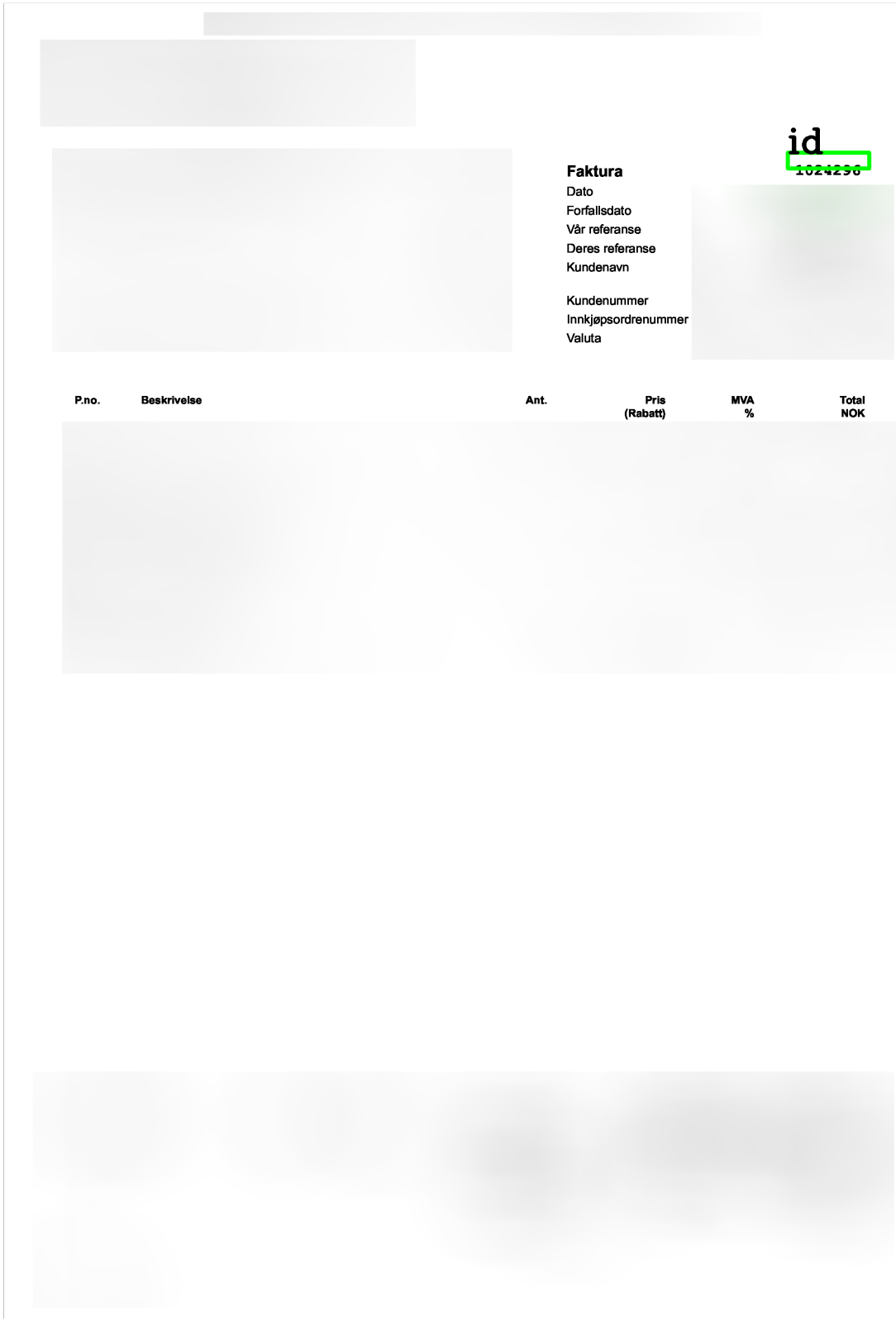
Kundenr.:
Deres ref.:
Vår ref.:

Sidenr.:

Beskrivelse	Antall	Pris	Netto beløp

Å betale:
Bankkonto:
Orgnr.:


id
2272
Faktura
Dato
Forfallsdato
Vår referanse
Deres referanse
Kundenavn
Innkjøpsordrenummer
Valuta
P.no.
Beskrivelse
Ant.
Pris
(Rabatt)
MVA
%
Total
NOK
Nettobeløp
Totalt MVA beløp
Beløp å betale


id
Faktura
Dato
Forfallsdato
Vår referanse
Deres referanse
Kundenavn
Kundenummer
Innkjøpsordrenummer
Valuta
P.no.
Beskrivelse
Ant.
Pris
(Rabatt)
MVA
%
Total
NOK

Faktura 338928

Fakturadato:
Bet.bet.:
Forfallsdato:
Kundenr.:
Ordrenr.:
Ordrebehandler:
Selger:
Merket:
Side:

Prod.nr	Beskrivelse	Antall:	Pris	Beløp	Mva. sats

id id

Kreditnota **100130**

Fakturadato:
Forfall:

Kundenr.:
Deres ref.:
Vår ref.:

Sidenr.:

Beskrivelse	Antall	Pris	Netto beløp

id id id

FAKTURA **247839**

Fakturadato:
Forfallsdato:
Deres referanse:
Kunde ID:
Vår referanse:
KID:

id

Faktura	Fakturanr	Dato	Tid	Kasserer	Side
	0000005602000025888				

id
1024295

Faktura
Dato
Forfallsdato
Vår referanse
Deres referanse
Kundenavn

Kundenummer
Innkjøpsordrenummer
Valuta

P.no.	Beskrivelse	Ant.	Pris (Rabatt)	MVA %	Total NOK

id

Faktura

Dato

Forfallsdato

Vår referanse

Deres referanse

Kundenavn

Kundenummer

Innkjøpsordrenummer

Valuta

P.no.	Beskrivelse	Ant.	Pris	MVA	Total

id
1025609

Faktura
Dato
Forfallsdato
Vår referanse
Deres referanse
Kundenavn

Kundenummer
Innkjøpsordrenummer
Valuta

P.no.	Beskrivelse	Ant.	Pris (Rabatt)	MVA %	Total NOK

Faktura 19042

Fakturadato:
Leveringsdato:
Forfallsdato:
Vår referanse:
Deres referanse:
Bilagsnr:
Bankgiro:
IBAN
SWIFT
Ref. fakturanr
Rekvisisjon

Faktura id 2273

Dato
Forfallsdato
Vår referanse
Deres referanse
Kundenavn

Innkjøpsordrenummer
Valuta

P.no.	Beskrivelse	Ant.	Pris (Rabatt)	MVA %	Total NOK

Telefon:
Foretaksregisteret
Bankkonto:
IBIC/SWIFT:
IBAN:
Kontakt:
Web:

id

FAKTURA

Fakturanummer 1002115109
Fakturadato
Forfallsdato
Å betale
Bankkonto:
KID
Kundenummer
Registreringsnr.

Giro F68-1 OpusCapita 16.05.2018 - 04:03:55