

Arnkværn, Brage
Schoeler, Sigurd

FinanceDoc2JSON: Parsing and structuring invoices and other financial documents with deep learning

Bacheloroppgave i Dataingeniør

Veileder: Tom Røise

Mai 2021

Arnkværn, Brage
Schoeler, Sigurd

FinanceDoc2JSON: Parsing and structuring invoices and other financial documents with deep learning

Bacheloroppgave i Dataingeniør
Veileder: Tom Røise
Mai 2021

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk



Kunnskap for en bedre verden

Abstract

This thesis documents our process of getting a machine learning model to understand financial documents and parse them to JSON. We created a dataset and evaluated multiple models, but ended up with YoloV3 to locate the fields. The final model finds the correct fields 57.56% of times on our test set.

Sammendrag

Denne oppgaven dokumenterer prosessen vår med å få en maskinlæringsmodell til å forstå finansielle dokumenter og tolke dem til JSON. Vi opprettet et datasett og evaluerte flere modeller, men endte opp med YoloV3 modellen for å finne feltene. Den endelige modellen finner riktige felter 57.56% ganger på testsettet vårt.

Preface

Thanks to Finexa for giving us such an interesting task and dataset and Tom Røise for guidance on the project.

Contents

Abstract	iii
Sammendrag	iv
Preface	v
Contents	vi
Figures	ix
Tables	xi
1 Introduction	1
1.1 Background and motivation	1
1.2 Finexa	2
1.3 Why is this a challenging task?	2
1.4 Project boundary and scope	3
2 Fundamentals	5
2.1 Field validation	5
2.1.1 Ambiguous fields	6
2.2 Machine learning	7
2.2.1 The basics of a neural network	7
2.2.2 CNN	7
3 Software 2.0 - Data-driven programming	10
3.1 Why AlexNet was a breakthrough	11
3.2 The definition of a chair	11
3.2.1 The long tail	12
3.3 Labeling is iterative	12
3.4 In software 2.0 the dataset is the code	12
3.5 How we used the software 2.0 paradigm?	13
3.5.1 Fetching more hard samples	13
3.5.2 Build upon other models	14
3.5.3 Machine learning is primarily an infrastructure problem	14
4 Finexa dataset	17
4.1 Invoices with properties	17
4.2 Payment vouchers	17
4.3 Real life data and our dataset	18
4.4 Data distribution	18
4.4.1 Uncommon data	18
4.4.2 Uneven distributions	19

- 4.5 Automatically labelling with weak supervision 20
- 4.6 Payment vouchers 21
- 4.7 The final dataset 21
- 5 Methods and implementation 23**
 - 5.1 The model 23
 - 5.1.1 One model to locate and one model to read 23
 - 5.1.2 How we evaluated the models 23
 - 5.1.3 U-net 24
 - 5.1.4 Yolov3 24
 - 5.1.5 Other models 24
 - 5.2 The evolution of Yolov3 26
 - 5.3 The application on top of the model 27
 - 5.4 Model output pipeline 28
 - 5.4.1 Processing the image 28
 - 5.4.2 Yolo bounding box prediction 28
 - 5.4.3 Postprocessing of bounding box 28
 - 5.4.4 EasyOCR for extracting the text 29
 - 5.4.5 Field revolvers 29
 - 5.5 Testing 29
 - 5.5.1 Model health tracking 31
 - 5.6 Interacting with the model 31
 - 5.7 The importance of a good frontend 32
- 6 Implementation and results 35**
 - 6.1 Invoices 35
 - 6.1.1 Result of the latest model on a synthetic dataset 37
 - 6.1.2 Result of the latest model on a out of sample dataset 43
 - 6.1.3 How much augmentation can the model handle ? 49
 - 6.1.4 F1-score of our model on bounding box matching on invoices 51
 - 6.2 Payment vouchers 52
 - 6.3 Result of the latest model on a synthetic dataset 52
 - 6.3.1 Example 1 52
 - 6.3.2 Example 2 54
 - 6.3.3 Example 3 56
 - 6.3.4 F1-score of our model on payment vouchers 59
- 7 Overview of existing solutions 61**
 - 7.1 DocParser 61
 - 7.2 TripleTex 61
 - 7.2.1 Accuracy 62
 - 7.3 Nanonets 62
- 8 Discussion 63**
 - 8.1 Development method and process 63
 - 8.2 Deviation from original project plan 64
 - 8.3 Other research 64
 - 8.4 Improvements 64

8.5	Comparison to other solutions	65
8.5.1	Using Python as the language of the application	65
8.6	Stealing peoples job?	66
8.7	Authors relation to Finexa	66
8.8	Toolchain / Software tools used	66
8.8.1	PyMuPDF	66
8.8.2	EasyOCR	66
8.8.3	Hardware used	67
9	Conclusion	68
9.1	Result	68
	Bibliography	69
	Glossary	72
	Acronyms	73
A	Meeting logs	74
A.0.1	15.01.2021	74
A.0.2	08.03.2021	75
A.0.3	06.04.2021	75
A.0.4	06.05.2021	75
B	Timesheet	76
B.1	Short summary for each month	76
B.2	Division of time between tasks	77
B.3	Github activity	77
C	Assignment description	79
D	Project plan	82
E	Project agreement (Prosjektavtale)	98
F	Group guidelines	102

Figures

1.1	The lower image vaguely shows the text which is hidden on black background, which can still be selected and searched for. Rasterizing the image would yield a loss of information.	3
1.2	Examples of invoice templates in our dataset	4
2.1	Example of convolution layer, a figure from [8].	8
2.2	Example of max pooling layer, a figure from [9]	9
3.1	First convolution layer of AlexNet[12], figure from [12]	11
3.2	13figure.caption.12	
3.3	The "Tesla data engine" presented by Karpathy [14].	13
3.4	Our "data engine"	14
3.5	VGG architecture (Generated with https://github.com/HarisIqbal88/PlotNeuralNet)	15
3.6	Resnet architecture (From https://github.com/HarisIqbal88/PlotNeuralNet/issues/24)	15
3.7	Karpathy on model architecture in real life, figure from [13]	15
3.8	Figure from[20] for how most people view machine learning, and how it should not be solved.	16
4.1	Examples of synthetic invoice based on a real invoice template.	18
4.2	Example of synthetic payment voucher based on payment voucher template.	19
4.3	Uncommon and common field locations	19
4.4	Elbow method results. Y-axis is distance between cluster center (inertia), X-axis is cluster size.	20
4.5	Distributions of invoices templates with a cluster size of 35. Axis is most visible in a PDF reader. Y-axis is percentage and X-axis is cluster number.	20
5.1	U-net with many fields	25
5.2	Accuracy of the first time running Yolov3	26
5.3	Example output of Pix2Pix, figure from [24]	26
5.4	Loss of the bounding box for Yolov5	27
5.5	Flat recall curve for Yolov5	27

- 5.6 Average F1 score of all the fields. The first Yolov3 model vs the latest. Glamorous-shape-20 is name of the latest model. 28
- 5.7 Resolver 29
- 5.8 Account number validator 30
- 5.9 Example of the plots we generated with Weights & Biases (wandb) 31
- 5.10 Sudden drop in F1, looks like a bug has slipped in... 32
- 5.11 Example of api request and response 33
- 5.12 Our application frontend where the user can upload PDFs (Work in progress) 34
- 5.13 The page a user will be redirected to when having uploaded a document. The fields in the form to the right will be filled in by our machine learning model, through the GraphQL API. (Design prototype) 34

- 6.1 Error from example 1 37
- 6.2 Results of example 1 38
- 6.3 Error from example 2 39
- 6.4 Results of example 2 40
- 6.5 Results of example 3 42
- 6.6 Results of NTNU invoice 44
- 6.7 Results of the Telenor invoice 46
- 6.8 Results of the Komplet invoice 48
- 6.9 Augmentation 50
- 6.10 wandb logs from our last invoice model 51
- 6.11 Results of basic/medium payment voucher 52
- 6.13 Results of basic payment voucher 54
- 6.15 Results of hard payment voucher 57
- 6.17 wandb results from the latest bankvoucher model 60

- 8.1 Our Gantt chart at the end of the project. The screenshot is from our project management framework called ZenHub [26], which is integrated with GitHub's issues. 63

- B.1 A summary of hours worked per month combined for all group members 77
- B.2 The GIT commit activity 78

Tables

4.1	How much a five sorted cluster together contribute to the dataset. One can see that 5 classes contribute to over 50% of the dataset. . .	21
4.2	Field for different documents	22
6.1	Results on all the different invoice templates. Tested and validated by a human for all the 35 different templates.	36
6.2	Result matrix of Figure 6.2	37
6.3	Result matrix of Figure 6.4	39
6.4	Result matrix of Figure 6.5	41
6.5	Result matrix of Figure 6.6	43
6.6	Result matrix of Figure 6.7	45
6.7	Result matrix of Figure 6.8	47
8.1	Hardware used	67

Chapter 1

Introduction

1.1 Background and motivation

The task Finexa gave us was to read PDFs and pictures of **Invoice** (*Faktura*) and **Payment Voucher** (*Bankbilag*) to extract structured data from various fields that are given on the document.

An application that can extract info from invoices and bank documents has a lot of valuable applications. To name some:

- Integration with accounting systems where Finexa does not get invoice data, but only Portable document formats (PDFs).
- Validate data about an invoice to make sure the data received is correct. In other words an extra step of validation.
- Automatically process information of a given document, and use that in a user application for automatic pipelines by providing a general .

Our report documents the process of creating FinanceDoc2JSON, which is able to parse different finance related documents to a machine readable format. It is important that this information is reliable so and that there is no false information that gets imported, as the information will be acted upon and is used to send payment reminders. We will focus on getting a high accuracy on different invoice layouts, as seen in Figure 1.2, and we additionally validate the info our machine learning model outputs. This means that part of the task includes finding different ways of validating the fields that are on the invoice. See more in section 2.1.

We use EasyOCR to run Optical Character Recognition (OCR) on the image and get the locations of all the text, and one can believe the task is solved by this. But we have not identified the critical fields, such as invoice number, customer name, invoice amount, and the Norwegian Kundeidentifikasjonsnummer (KID). This is elaborated on in subsection 2.1.1. In chapter 5, we use a computer vision model to find the location of these fields, and extract the text from the location to find the value. In section 2.1 we talk about the validation and cleaning we do on the extracted text, so our system does not output incorrect data.

In chapter 3, we will put extra weight on the Software 2.0 paradigm, and how

we have used it to create a dataset consisting of 30 000 invoices. Finexa already has the json) data linked to each invoice PDF, and we use this data in section 4.5 to create a weakly supervised dataset.

In chapter 5 we go into choosing a machine learning model. We evaluate 5 different models, and end up using YoloV3, and continue in section 5.3 how we integrated the model into the rest of our application.

1.2 Finexa

Finexa is a invoice follow-up and debt collection agency, which helps their customers to get their Accounts Receivable paid on time¹. Finexa's data is sourced by their partner Izy Integrations, which retrieves the PDFs and corresponding JSON from the customer's accounting systems.

Finexa also gets payment vouchers from different banks. These files are PDFs that document a payment of an invoice. These also usually comes with structured data, but sometimes not.

The invoice and payment data is used in their **Accounts Receivable Management** (*Fakturaoppfølging*) service. If Finexa receives PDFs *without* structured data, their employees have to manually enter the information about the invoice into their system, which is a labor-intensive and error-prone process. Finance-Doc2JSON will help them with these issues.

1.3 Why is this a challenging task?

There are multiple challenges with finding key info in a document. challenges:

- Our solution should work on as many as possible file types.
- The PDFs specification is very long and difficult, and a PDFs can contain lots of different types of information. some of the things that are included in a PDFs are:
 - characters, words, lines, blocks
 - images
 - multiple pages
 - fonts
 - different color spaces
 - metadata
 - embedding arbitrary files
 - JavaScript scripts
 - may be scanned

Because of the complexities in the PDFs specification, there are not many solutions which support editing PDFs, and some are even paid to use this feature, like Adobe Acrobat Reader. The best python supported open source implementation

¹<https://finexa.no/>

that supports editing, rendering, text extraction, image extraction, rendering and converting to image we found is PyMuPDF[1]. A lot of the complexities of the PDFs specification has been removed by using this library and has enabled us to work programmatically with PDFs files.

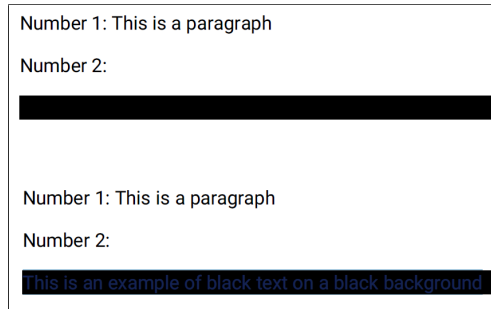


Figure 1.1: The lower image vaguely shows the text which is hidden on black background, which can still be selected and searched for. Rasterizing the image would yield a loss of information.

A PDFs file may be searchable. To simplify this matter, this is achieved by adding a text layer above the vector graphics. We illustrate this in Figure 1.1, where there is black text on black background. Using most PDF-viewers, like adobe acrobat or chrome's built-in text viewer, we can select this text and copy it to the clipboard, even if it is not visible. This is because of the text layer behind the graphics. This also means we can add text to the text layer which is not visible in the PDF. Because of this complexity, it is hard to find specific information which is in the document, for example the invoice number. There are 2 solutions to this, we can extract all of the text which is in the PDFs to a text file, and find the info in this text file. The other solution is to convert the page into an image and analyze this image with computer vision techniques. When converting from PDFs to image, we lose the information about what text is at the location in the PDF. We ended up using both of these techniques in FinanceDoc2JSON.

1.4 Project boundary and scope

We will focus on making the system stable for invoice, payment voucher, and add validation and approval of the data, but we will also discuss ways of adding support for other types of documents, such as receipts.

Because OCR in general is looked at as a solved problem, we will focus on creating a model that is able to locate the relevant fields in a good way. In other words our model should output a mask so we can extract only the relevant field and give the image mask to an OCR model and have it extract the text.

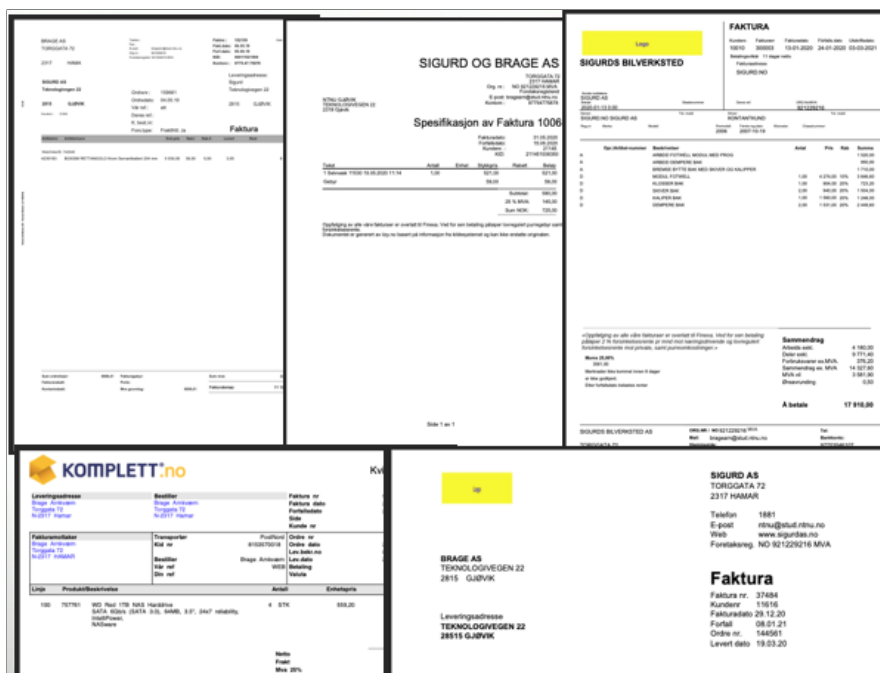


Figure 1.2: Examples of invoice templates in our dataset

Chapter 2

Fundamentals

2.1 Field validation

The data in financial documents are connected to the real world and is governed by many different properties:

- Tax laws and other laws about what info is required to be on an invoice such as supplier inf and information about how to pay the invoice. We can make the following assertions: Gross amount, net amount, Tax amount, invoice line sums and a number of units should be formatted as numbers. They should also always have currency specified.
- The sum of the invoices lines should always sum to the total amount on the invoice.
- Dates should be valid dates and Invoice date field is chronologically before the payment date field.
- KIDs, organization numbers, social security numbers, and bank account numbers have built in validation by adding a check digit to the end, which is usually calculated using some form of the Luhn algorithm¹, using either modulus 10 or 11. The control digit is added to prevent human errors when manually entering numbers when transferring payments. This works by calculating a weighted sum of every number, but the last modulus 10 or 11. If the weighted sum equals the control digit, the number is valid.² 1
- If the recipient is a company, the organization number, company name, and company address must exist in Brønnøysundregistrene³, and these values can be cross-validated between each other.
- There are only 4 different VAT-rates in Norway.

This is not an exhaustive list, and there are many implicit things that invoices do. With enough data, a machine learning model should learn some of these, instead of explicitly coding each feature.

¹https://en.wikipedia.org/wiki/Luhn_algorithm

²<http://www.lefdal.cc/div/mod11-sjekk.php>

³<https://www.brreg.no/>

```

1 class Modulus:
2     def __init__(self, modulus: int, weights: List[int], operation:
      ↳ Operations = None) -> None:
3         self.modulus = modulus
4         self.weights = weights
5         self.operation = operation
6
7     def calculate_check_digit(self, str_without_check_digit: str) ->
      ↳ int:
8         str_without_check_digit_list: List[int] = list(map(int,
      ↳ list(str_without_check_digit)))
9         self.weights += len(str_without_check_digit) //
      ↳ len(self.weights) * self.weights
10        self.weights = self.weights[: len(str_without_check_digit)]
11        zipped = zip(self.weights, str_without_check_digit_list)
12
13        operation = lambda x: x
14        if self.operation == Operations.SUM:
15            operation = lambda x: sum(list(map(int, list(str(x)))))
16        weighted = sum([operation(x * y) for x, y in
      ↳ reversed(list(zipped))])
17
18        remainder = weighted % self.modulus
19        if remainder == 0:
20            return 0
21        return self.modulus - remainder

```

Listing 1: Modulus class

2.1.1 Ambiguous fields

Most fields are ambiguous and can not be fully validated based only on the value. Most invoices have a date and due date field, but some may also have a registration and delivery date. There is no direct way to know if a date is an invoice date, payment due date, registration date, or delivery date. The same is true for supplier name, buyer name and addresses of both. This is usually pretty apparent from the invoice layout and the position of the text, but there is no real way for us to validate these fields only based on the output text value.

There is also fields you can indirectly validate like the invoice amount. By applying the rule that the invoice line amounts should sum to the invoice total, and can therefore be cross-validated by summing up the invoice lines, and rejecting the value if it is far off. This is not currently implemented, but is an idea for further improving the output of the application.

Extracting Structured Data from Templatic Documents extended this idea by

using a binary classifier to score the candidates based on how likely it is that the value corresponds to its field. They also used information about nearby fields, and the candidates position.

Even when a machine learning model uses information about layout and design, there might still be ambiguity, as our final model mainly looks at the actual value to determine if the field is correct. The best way to solve for this is to train the model on a great variance of data. Which is what we tried to do in chapter 5.

2.2 Machine learning

We assume the reader already knows the basics of machine learning, and the key concepts behind how it works. We will give a short introduction, but if the reader knows the basics of deep learning and that they are universal function approximate trained with gradient decent they can skip this section. If the reader wants to learn more they should look at [2][3][4].

2.2.1 The basics of a neural network

The building blocks of machine learning is linear algebra and a bit of calculus. By applying matrices with matrix multiplication and an error function (also called loss function) with an optimizer you have a basic neural network. The matrices store what is commonly called the weights, which we can say is analogous to many switchers, but is often portrayed as neurons. When many of these switches are combined, you can achieve very complex behavior.

When you try to train a basic neural network, you will run matrix multiplications against the layers in a "feed forward" fashion. To normalize the input between the layers one usually apply an activation function (which is just a non-linear function) on the results of each layer. This is done to prevent the results from having numeric overflows. Sigmoid⁴ was a popular activation function, but nowadays most people use ReLu⁵. You start with running matrix multiplication on the input and the first layer, then the results of that against gets put into an activation function and matrix multiplication is run against the next layer, this is done for all remaining layers. Then you apply the error function against the output and expected output. Then the weights are updated based on how much they contributed to the final error, by backpropagating the error[5]. This can all be seen in 2. Layers, error functions, and optimizers all vary but most build upon the concept described above.

2.2.2 CNN

Convolution neural nets (CNN) are neural nets that contain largely convolution layers. Which are layers that perform a set of convolution operations on the input

⁴ $\frac{1}{1+\exp-x}$
⁵ $\max(0, x)$

```

1 import numpy as np
2 features = np.array([ [0,0,1],[0,1,1],[1,0,1],[1,1,1] ])
3 labels = np.array([[0,1,1,0]]).T
4
5 initialize_weights = lambda x, y: 2 * np.random.random((x,y)) - 1
6 weights_0 = initialize_weights(3,4)
7 weights_1 = initialize_weights(4,1)
8 activation = lambda x, deriv=False: x*(1-x) if deriv else 1/(1 +
  ↪ np.exp(-x))
9 for _ in range(60000):
10     output_layer_1 = activation(np.dot(features, weights_0))
11     output_layer_2 = activation(np.dot(output_layer_1, weights_1))
12     layer_2_delta = (labels - output_layer_2) *
  ↪ activation(output_layer_2, deriv=True)
13     layer_1_delta = layer_2_delta.dot(weights_1.T) *
  ↪ activation(output_layer_1, deriv=True)
14     weights_1 += output_layer_1.T.dot(layer_2_delta)
15     weights_0 += features.T.dot(layer_1_delta)

```

Listing 2: How a basic neural network can be made based off code from [6]

with a filter. This is visualized in Figure 2.1. The filter is applied to the input data, and the sum is then stored as an output feature. The filter is other words analogous to the weights in a standard neural network. In other words, the filter is parameters that are optimized.

What is interesting is that the filter is able to learn filters that previously has been handcrafted by humans as we will discuss in section 3.1.

There is also a Polling layer that will downsample the input, max pooling will for instance output the max value inside a window as illustrated by Figure 2.2.

We advice the reader to look at *CS231n Convolutional Neural Networks for Visual Recognition* [7]

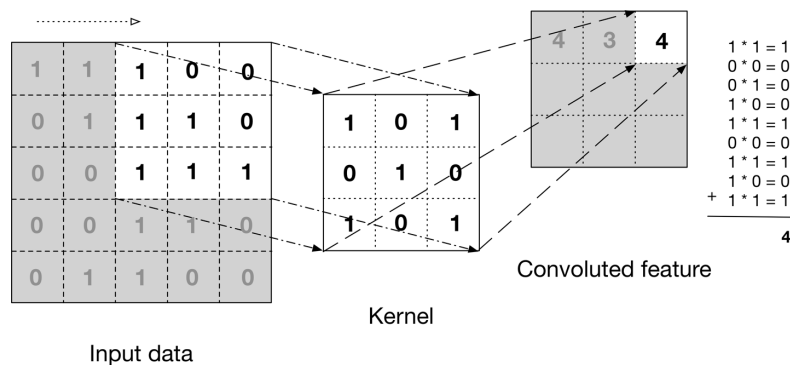


Figure 2.1: Example of convolution layer, a figure from [8]

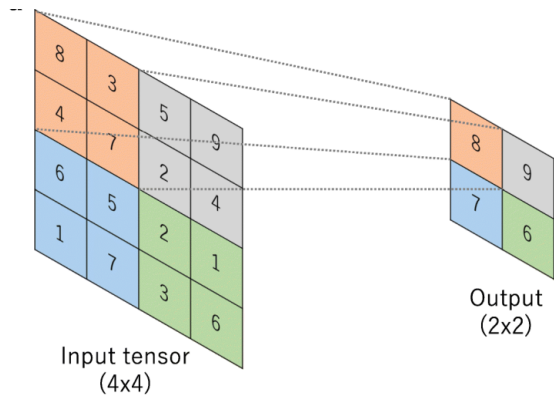


Figure 2.2: Example of max pooling layer, a figure from [9]

Chapter 3

Software 2.0 - Data-driven programming

Software 2.0 is a term for a new software paradigm where the program is learned by a lot of data. The term was popularized by Andrej Karpathy in [10], most famous for leading the development of the Tesla Autopilot system[11]. The paradigm involves letting optimization algorithms define the program “code” by looking at a lot of data. Most of the development is also centered around helping these learned functions perform as good as possible by giving them a variety of input and continually finding edge cases to improve on.

Why should we use data to learn a function instead of going with a rule-based system? For this, we have to look at some machine learning history. For a long time all features were handcrafted around different rules. In computer vision, Scale-invariant feature transform (SIFT) and Hough transform were heavily used ¹². There are many reasons for why this was the case for so long:

- Compute used to be hard to find. Today compute is available, and it’s cheaper than ever. Big neural networks are possible to run on a gaming GPU, or just rent it on AWS ³.
- Previously there was no good common libraries. Nowadays, many of the biggest companies have published their own open source machine learning library ^{4 5}.
- Many had also lost faith in deep learning ⁶.

¹https://image-net.org/static_files/files/ILSVRC2010_NEC-UIUC.pdf

²<https://image-net.org/challenges/LSVRC/2010/>

³<https://docs.aws.amazon.com/dlami/latest/devguide/gpu.html>

⁴<https://github.com/pytorch/pytorch>

⁵<https://github.com/tensorflow/tensorflow>

⁶https://en.wikipedia.org/wiki/AI_winter section "Developments post-AI winter"

3.1 Why AlexNet was a breakthrough

Classifying an image is no easy task for computers. The computer does not have eyes as we do, so we have to describe the image with numbers in a matrix. To compress the image down and make it more “readable” for the computer, we use feature extractors to only get out the relevant information. The question now is, what are good features and relevant information? That is not an easy question to answer. For a long time, computer vision practitioners looked at edge filters, image gradients, different image scales, etc. But these features are not enough to carry all of the information in an image.

The way most deep learning architectures works is to learn the feature extractors. There has been a lot of research on what features these models pick up on⁷⁸⁹. The original AlexNet paper[12] even included a visualisation of the first convolution layer as seen in Figure 3.1. Interestingly, the first layer chooses many filters humans commonly have handcrafted, like edge detection. The big breakthrough here is the fact that feature engineering has now become obsolete, since the model is clearly able to learn the filters previously handcrafted and it beat all other models.



Figure 3.1: First convolution layer of AlexNet[12], figure from [12]

3.2 The definition of a chair

If we look at a standard chair, it usually has a back and four legs. Using that as a heuristic and handcrafted feature, you would not be able to classify a barstool as a chair. You can continue to add new heuristics for the barstool, but then what about a bean bag ? That can also can be used as a chair. You can continue to do this, but there will always be some part of the specifications of a chair missing. Instead, defining a chair as what a neural network says a chair is after seeing many thousands of rich variants of chairs. You will get a much better definition.

⁷<https://distill.pub/2017/feature-visualization/>

⁸<https://distill.pub/2021/multimodal-neurons/>

⁹<https://arxiv.org/abs/1904.08939>

By not restricting the system to human definitions of what a char looks like, one can see much greater creativity emerge from the model. There have been quite a few results in recent years showing that by not restricting the algorithms to expert knowledge, they will surprise us with new ideas and creativity. A more recent example of this is Alpha GO playing move 37 and 78 against Lee Sedol¹⁰.

3.2.1 The long tail

Because of the many invoice design templates as can be seen Figure 1.2, how would one be able to know what an invoice number look like? In the software 2.0 paradigm, the definition of an invoice number is what the neural network tells us an invoice number looks like. Given that the neural network will see the representation of many thousands of invoice numbers with location and other fields close by, it will be able to generate a much better definition of how an invoice number than we can Figure 3.2.

3.3 Labeling is iterative

The dataset is never finished. Which is now how most academic research sees it. In academic research, you are usually given a dataset like MNIST¹¹ or Image-Net¹² and the task is to create the best model. The dataset does not change much here.

Our task is different, the dataset will change over time. New accounting system will be created and the invoice templates will change. The model should therefore continually be trained on new data and especially new harder samples.

By integrating the model with a user interface, the model could also automatically get the new labels from having a human in the loop. If it a field wrong, the user will correct the mistake, and the model becomes better by automatically adding the new label to the dataset and shipping a new model at night.

It is also worth noting that we noticed new templates had been added from when we started the project, and when we were closed to finish. Having good integration for this helps boost performance.

3.4 In software 2.0 the dataset is the code

The machine learning model is a function approximator. It learns to map an input to a given output, the basis of a machine learning problem. To do this well, the dataset has to be divorces, ideally big, and “clean”. The dataset is what defines the boundaries of the system and also the behaviors. Karpathy’s analogy is that instead of compiling source code files into a binary, a neural network will compile the dataset into a binary.

¹⁰<https://www.quora.com/What-was-the-significance-of-move-37-and-move-78-in-Go-AlphaGo-versus-Lee-Sedol>

¹¹<http://yann.lecun.com/exdb/mnist/>

¹²<https://www.image-net.org/>

As discussed in his talk *Building the Software 2.0 Stack (Andrej Karpathy)*[13] the software 2.0 IDE is not here yet. We therefore had to create tools ourselves to combat some of the problems for visualization and finding edge cases section 3.5 subsection 4.4.2.



Figure 3.2: Karpathys Tweet saying that a Software 2.0 solution is better than your handcrafted features ^a

^a<https://twitter.com/karpathy/status/893576281375219712>

3.5 How we used the software 2.0 paradigm?

3.5.1 Fetching more hard samples

One thing we used a lot was looking at the validation loss to find images that was hard and or easy. This is a common thing to do to make sure the model and dataset behave correctly. Figure 3.2

Karpathy also discussed the “Tesla data engine”[14] as visualized in Figure 3.3. We also built upon this idea with the combination of validation loss. By having code for “continuous improvement” which would get more data from a source if it’s has a high loss. By fetching more samples when we find a hard sample Figure 3.4 we would make the model more robust to hard samples.

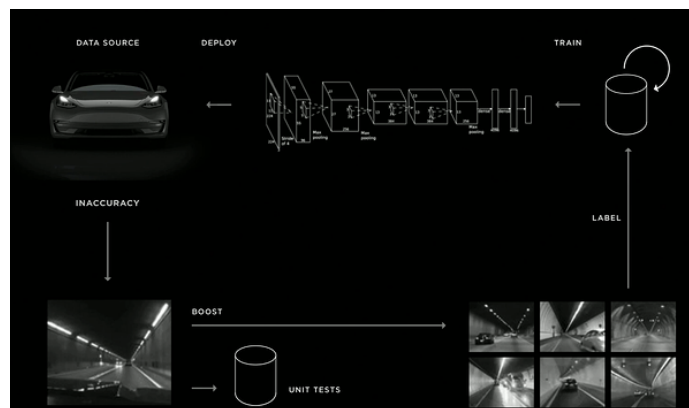


Figure 3.3: The ”Tesla data engine” presented by Karpathy [14]

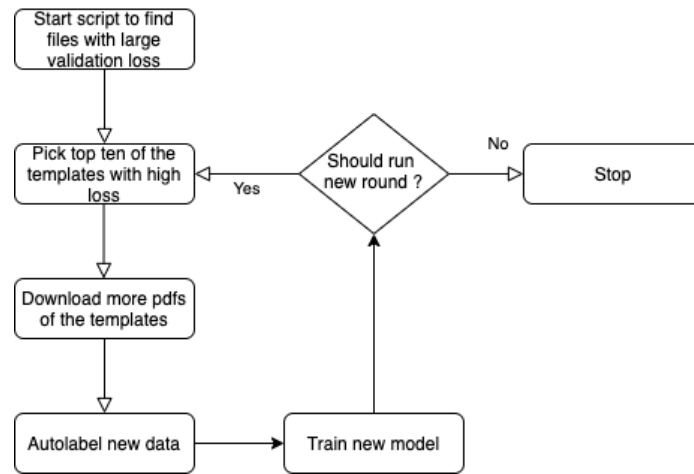


Figure 3.4: Our “data engine”

3.5.2 Build upon other models

We have also built upon other models. For instance, a lot of the dataset comes from one cluster as discussed in subsection 4.4.2. We used a k-means model to make it possible to optimize our primary model further. The k-means models were used to preprocess the dataset. For instance, it was used to generate the weights for the weighted dataloader as mentioned in section 5.2.

3.5.3 Machine learning is primarily an infrastructure problem

If one looks at the algorithms used today, most of the core algorithms are over a decade old. The core building blocks of AlexNet was algorithms that were over a decade old [15] and [5]. These algorithms are also the once used in the models we used. There are of course been some upgrades of the building blocks and introduction of new like [16] and [17], but even the architecture of common machine learning models [18] and [19] builds upon the CNN architecture idea that was introduced in [15]. There are of course, some tweaks, for instance with Resnet they have added the ability to skip connections between layers. However, the layers are still primarily CNN.

One can also see this with a common image shared by Karpathy Figure 3.7. Even though most people unfamiliar with machine learning thinks that most of the time is spent playing with the model. If you plan to solve a problem, then infrastructure is what needs to be solved first. Most of the performance gain is not in tweaking the model, but in creating a solid dataset. This is also said by Googles director of research Peter Norvig ”We don’t have better algorithms than anyone else. We just have more data.”¹³ and “Instead of trying to be clever with an new algorithm if you gather more data the worst published algorithm will beat

¹³<https://youtu.be/ql623nyCdKE?t=341>

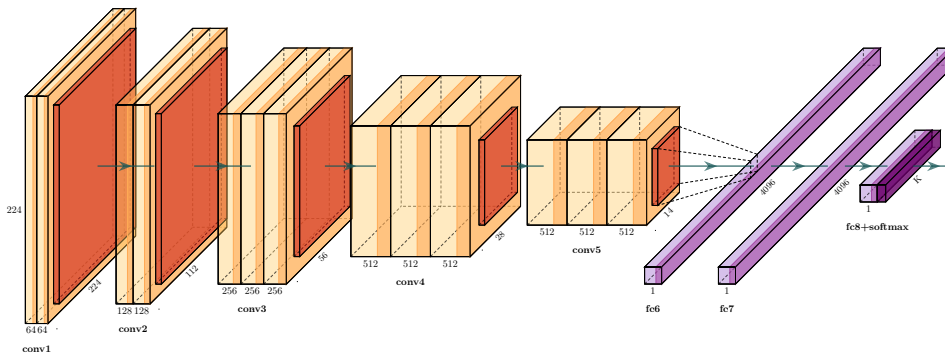


Figure 3.5: VGG architecture
(Generated with <https://github.com/HarisIqbal88/PlotNeuralNet>)

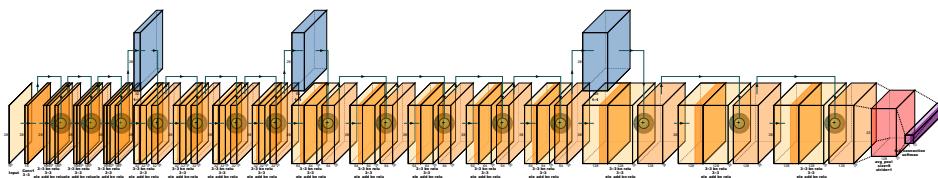


Figure 3.6: Resnet architecture
(From <https://github.com/HarisIqbal88/PlotNeuralNet/issues/24>)

the best”¹⁴. However to be able to fetch more data, handle incoming data and integrate new data, one needs good infrastructure.

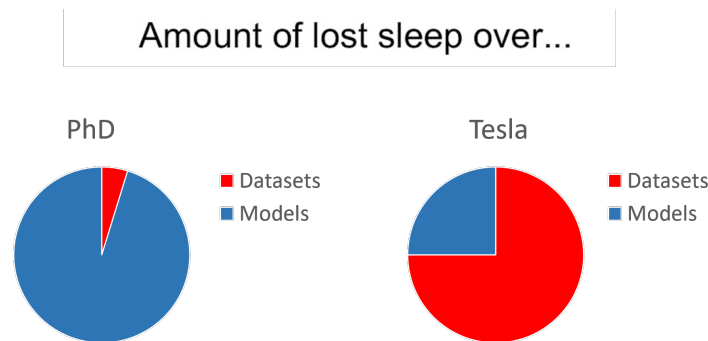


Figure 3.7: Karpathy on model architecture in real life, figure from [13]

So what is meant by machine learning being an infrastructure problem? To train a model which solves a problem it’s a lot about the Long tail subsection 3.2.1. The main problem to solve is the many edge cases. To do that, one needs the infrastructure to find the hard samples, fetch more hard samples, automatically label that, and then be able to release a new model. Many of these are ideas are

¹⁴<https://youtu.be/yvDCzhhjYWs?t=1031>

also discussed in the talk by Karpathy at “Tesla Autonomy Day”[14].



Figure 3.8: Figure from[20] for how most people view machine learning, and how it should not be solved.

Chapter 4

Finexa dataset

The dataset which Finexa provided contained 1.2 million(Appendix C) PDFs of invoices with corresponding structured JavaScript Object Notations (JSONs) file with the different invoice properties, like invoice number, kid, debtor and creditor names and addresses, etc. This data ranged over multiple years, and contains various invoice layouts. This gives us a rich dataset with high variance.

4.1 Invoices with properties

The JSON file looks very similar to what we try to produce 3. Keys for describing different properties of the document, and values for those properties. One can also see that invoices come in different shapes and forms in Figure 4.1

```
1 {
2   "kid": null,
3   "date": "2020-12-28",
4   "due_date": "2021-01-07",
5   "principal_amount": "6272,00",
6   "customer_number": "1149248",
7   "invoice_account_number": "15062160508",
8   "invoice_number": "10004"
9 }
```

Listing 3: Example JSON for Figure 4.1

4.2 Payment vouchers

The dataset also had PDFs of payments vouchers with a few different layouts, as can be seen Figure 4.2. These did not have a corresponding JSON file so they had

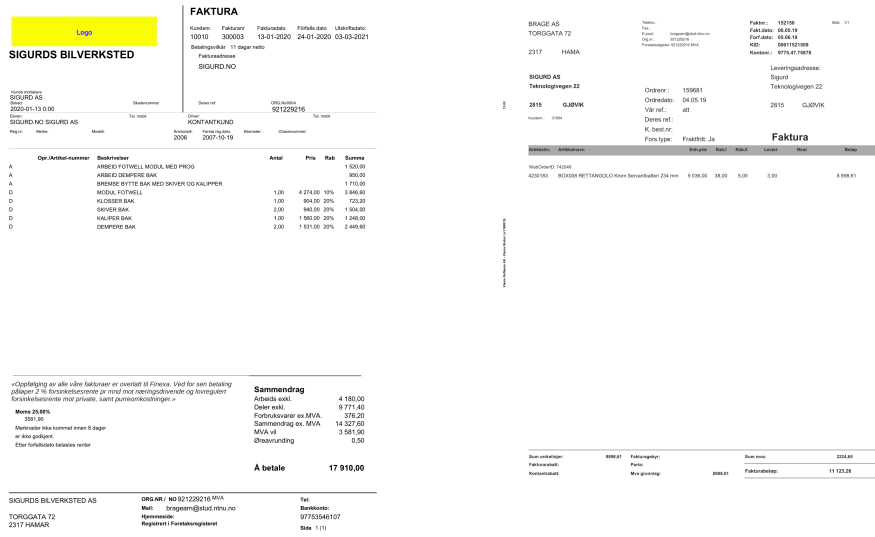


Figure 4.1: Examples of synthetic invoice based on a real invoice template.

to be labeled manually. The dataset we ended up using is therefore smaller, but demonstrates that our model is able to learn to extract key fields here as well.

4.3 Real life data and our dataset

A large proportion of the PDFs in the dataset are digitally created, and are therefore of high quality. However we cannot assume that the input we get into FinanceDoc2JSON to be of such high quality, and might be scanned, taken image of and be degraded. We will in the section 5.2 discuss how we further make the model be able to accurately handle degraded and badly captured invoices.

4.4 Data distribution

4.4.1 Uncommon data

There are, however some critical values that are uncommon in the dataset. Most of the invoices are created issued in Norway, and use NOK as the currency. This means we will not be able to handle different languages and currencies such as EUR or USD. To overcome these shortcomings in the dataset, the ideal solution would be to generate PDF with custom values and layouts, such as has been done in Attivissimo *et al.* [21]. Because of the PDF internal structure of PDFs this is a bit hard¹, and is not a task we had time to look at deeply.

There is also some structure that is uncommon in the data. That can be seen Figure 4.3, we found only one template with this kind of structure. So for the

¹<https://github.com/pymupdf/PyMuPDF/issues/257>

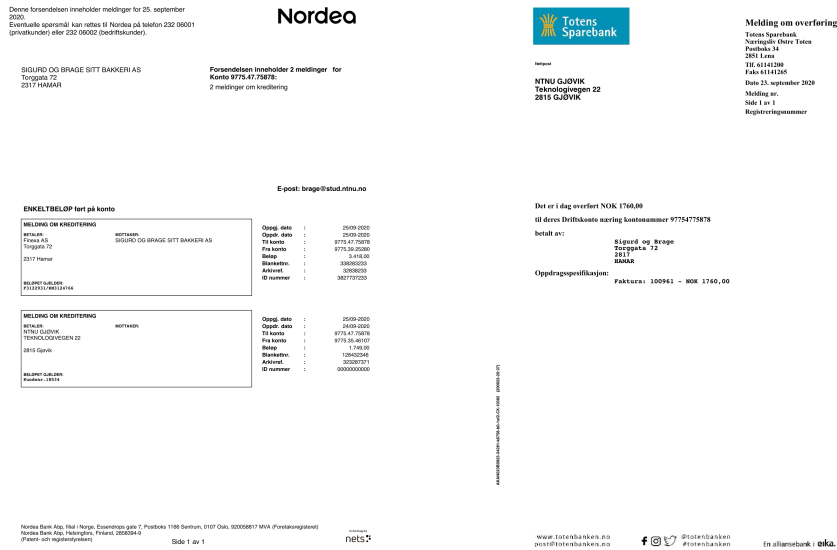


Figure 4.2: Example of synthetic payment voucher based on payment voucher template.



(a) Small percentage of the data has the fields value being below the field descriptor

(b) Most of the field descriptors and value goes horizontally

Figure 4.3: Uncommon and common field locations

model to be able to deal with this we had to apply some techniques as described in section 5.2, but there is still a way to go Figure 6.4.

4.4.2 Uneven distributions

There was a large part of the dataset that was based on the same invoice template. This can be seen by the Figure 4.4 and Figure 4.5. Because of this, we added weights to the dataset, so the model does not think most invoices are like the most common template in the dataset. This reduces overfitting and model bias.

While there is no clear elbow here like there are with some other data distributions, one can see that the curves start to have an elbow like shape near a cluster size of "35"(when looking at the results, this was very reasonable). When we tried to cluster based on that, we got the following dataset distribution Figure 4.5. As one can see some templates are much more used in the dataset. If we sort the

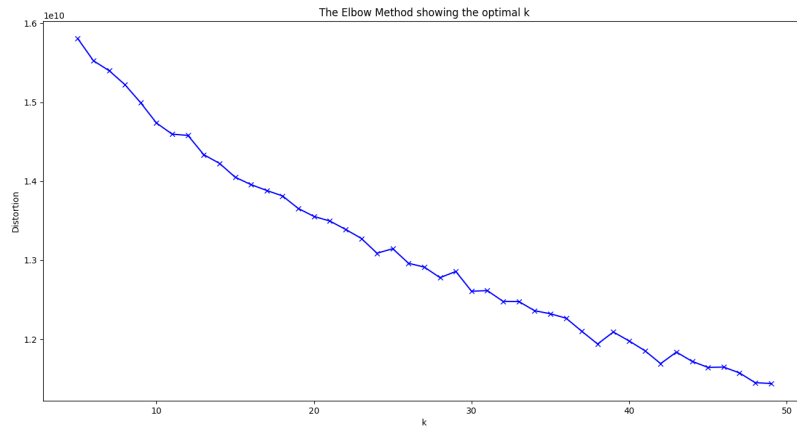


Figure 4.4: Elbow method results. Y-axis is distance between cluster center (inertia), X-axis is cluster size.

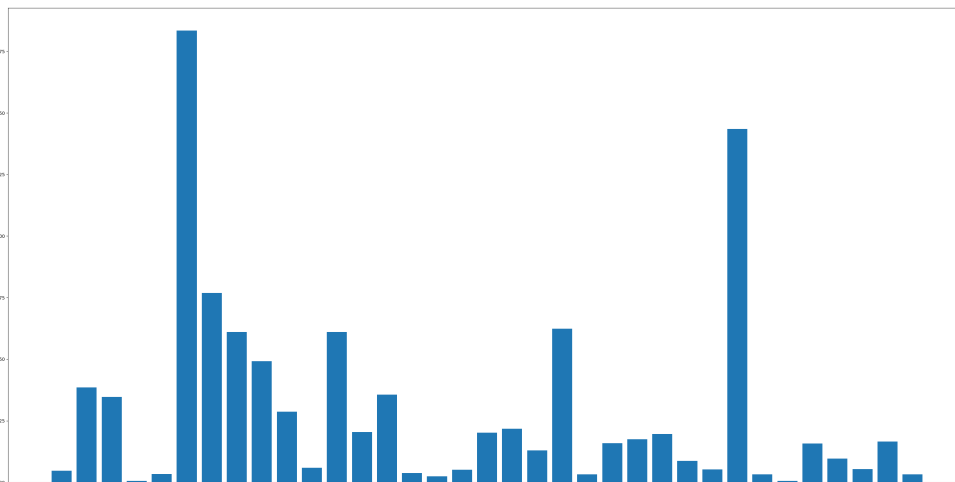


Figure 4.5: Distributions of invoices templates with a cluster size of 35. Axis is most visible in a PDF reader. Y-axis is percentage and X-axis is cluster number.

cluster size of each invoice template and sum the value of a chunk of size 5, we can clearly see that 5 invoice templates account for more than 50% of the dataset, as can be seen Table 4.1.

4.5 Automatically labelling with weak supervision

Because we knew the values of the properties of the invoices, we could create a script to find them and thereby label the dataset automatically.

By first locating the keywords for a property and then searching for the value

Chunk	Size (%)
0	0.01
1	0.02
2	0.03
3	0.08
4	0.11
5	0.22
6	0.53

Table 4.1: How much a five sorted cluster together contribute to the dataset. One can see that 5 classes contribute to over 50% of the dataset.

close to the keyword, we automatically labeled a large part of the dataset. Some PDFs had no actual text, but had an image inside the PDF so we first had to run OCR on the input. Some rules did not work well with certain invoice templates. We therefore also added custom rules for some designs. One can view this as an uncompressed way of finding the different invoices fields in a template. When the model trains on this it will compress this down to a weight and generalize it even more like discussed in section 3.4. It is also important to remember that this “hardcoded labeler” could not be used in production as the only reason it works is because we know the value we are searching for (because of the JSON file provided). This automatically labeling is close to what was done in [22].

4.6 Payment vouchers

The payment voucher data was not labeled, so we had to label it our self. We used coco-annotator for this ², and created a bounding box for the fields we ended up using.

4.7 The final dataset

The final dataset we trained on had 30 000 invoices and 500 payment vouchers. The fields we tried to predict where the once you can see in Table 4.2. We trained on an image size of 1030x1030 for the best quality.

Since we trained on a small subset (3% of the entire dataset), one could get even better performance by training on more data. However, since one epoch took around 15 minutes, and we wanted to train for ideally one 100 it took almost one day of training time to achieve 100 epochs. Therefore we could not have a much bigger dataset. The reason for the payment voucher dataset being so small was that we had to hand label the dataset ourselves, and labeling took some time if one wanted to be precise.

²<https://github.com/jsbroks/coco-annotator>

Field	Invoice	Payment voucher
Principal amount	✓	✓
Date	✓	✓
Due date	✓	
Customer number	✓	
KID	✓	
Invoice account number	✓	
From account number		✓
To account number		✓
Payer		✓
Creditor		✓

Table 4.2: Field for different documents

Chapter 5

Methods and implementation

5.1 The model

The model is at heart of this application. However as discussed in chapter 3 we believed a better model could always be changed later. Therefore we decided to try a few models, but not hold on too long to them if they did not work.

We looked at several segmentation models and object detection models before ending up with using Yolov3. By first selecting some popular segmentation models ¹² and some popular object detection models ³ we began to evaluate them.

5.1.1 One model to locate and one model to read

Because OCR is looked at as a solved problem for computer generated text with many open source programs ⁴⁵, we wanted to spend most of our time on making a model that is good at locating the relevant fields. We therefore created our own model for locating the field, and used EasyOCR on the field once it was located to extract the text.

5.1.2 How we evaluated the models

In the beginning, we evaluated the models by looking at how well both models worked together, which means the output of our model + EasyOCR. This is a valid metric, but since our model could output a perfect bounding box and then EasyOCR could misread a character, the metric had some flaws since our model cannot optimize for the output of EasyOCR. We therefore later looked at the F1 score for the bounding boxes, when training and testing. Then we ran a separate evaluation pipeline to test both models together.

¹<https://neptune.ai/blog/image-segmentation-in-2020>

²<https://medium.com/intel-student-ambassadors/segmentation-using-generative-adversarial-networks-80a161cf3>

³<https://arxiv.org/pdf/1807.05511.pdf>

⁴<https://github.com/tesseract-ocr/tesseract>

⁵<https://github.com/JaidedAI/EasyOCR>

5.1.3 U-net

U-net[23] was designed for dealing with biomedical images and is very popular for image segmentation⁶⁷. We therefore decided to try it for our task.

We built upon an implementation by Aman Arora⁸. Going for an autoencoder architecture. There was also mentioned an GAN architecture design in the paper, but we did not try this.

The model worked fairly well when training to predict a mask for a single field reaching an accuracy⁹ of 83.3% at it's best. However the results when down when we scaled up the model to predict new fields, staying below 50% accuracy.

When we investigated why, we could see that the U-net mask became less clear as more fields were added, as shown in Figure 5.1a. Applying some post-processing would probably help, but we did not try this because we would like to do the least amount of postprocessing of the mask. So we tried to see if there were other models we could try who did not have this problem and would return to U-net if nothing else was found.

5.1.4 Yolov3

Yolov3¹⁰¹¹ was an incremental improvement over the original implementation, but with less controversy¹². We built upon a version¹³ with a smaller codebase this time to make it easier to debug.

As with the other models, we had to do modifications to make the model trainable with our dataset, but we quickly got accuracy curves that looked like this Figure 5.2 as one can see the accuracy is pointing in the right direction.

5.1.5 Other models

Pix2Pix

Pix2Pix[24] is a generative model for producing image to image translation. Since this is a GAN, it would require quite a bit more compute than the other models we would be testing, but since they had demonstrated good results in the paper, it can be seen Figure 5.3, we wanted to take a look. If the extra compute could translate into good results. That did not happen. While the loss went down, the results were rubbish with an accuracy of 0%. Since an epoch would take close to an hour, we decided to pull the plug after 20 epochs. Note that this was tested

⁶<https://paperswithcode.com/paper/u-net-convolutional-networks-for-biomedical>

⁷https://scholar.google.de/scholar?oi=bibs&hl=en&cites=10845403114495995712&as_sdt=5

⁸<https://amaarora.github.io/2020/09/13/unet.html>

⁹Accuracy is here defined as the final output of both models combined, and all other usage of the word accuracy references to this. Unless otherwise specified.

¹⁰<https://github.com/eriklindernoren/PyTorch-YOLOv3>

¹¹<https://arxiv.org/abs/1804.02767>

¹²<https://blog.roboflow.com/yolov4-versus-yolov5/>

¹³<https://github.com/eriklindernoren/PyTorch-YOLOv3>

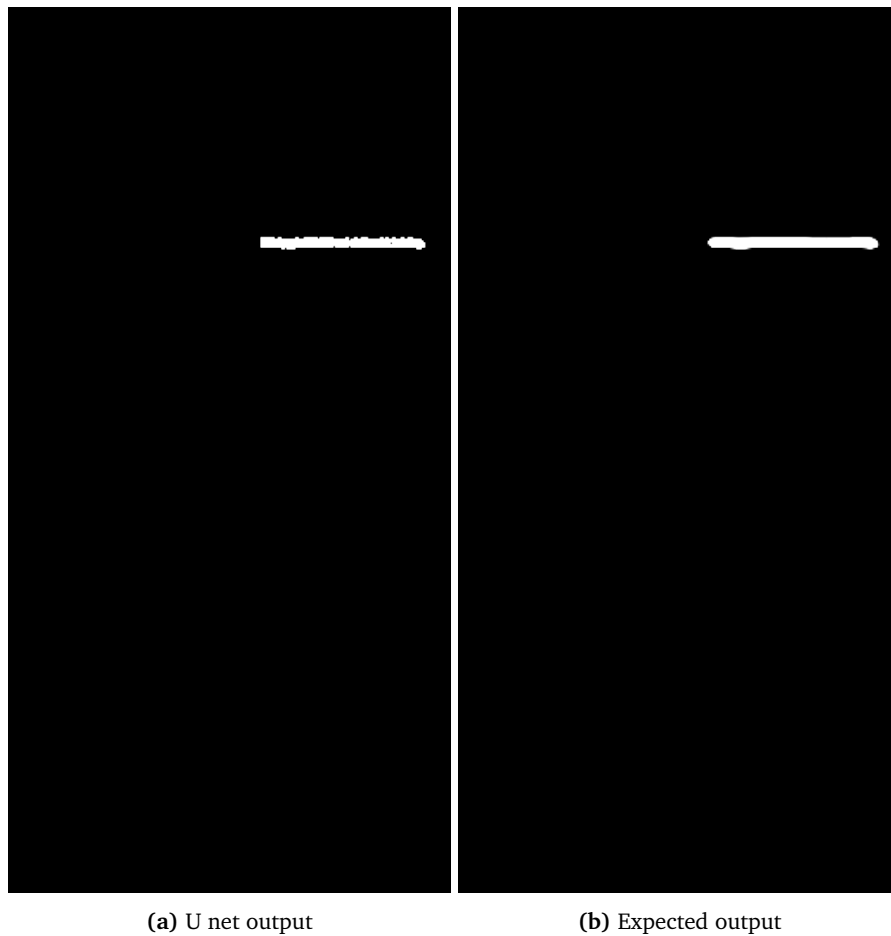


Figure 5.1: U-net with many fields

before upgrading the GPU as discussed in subsection 8.8.3, so it might be worth investigating with a powerful enough GPU.

Mask R-CNN

Mask R-CNN¹⁴ is a model published a few years back, and that then did well for several benchmarks tasks.

We had some problems with this model and was not able to produce any good results. We build upon a prebuild pytorch model¹⁵. Getting a lot of out of memory problems with high resolution images and not good results on image scaled down. This was before we upgraded to a new GPU as mentioned in subsection 8.8.3, so these problems might have been resolved with the new GPU.

It was mentioned in “Invoice 2 Vec: Creating AI to Read Documents - Mark

¹⁴<https://arxiv.org/pdf/1703.06870.pdf>

¹⁵https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html

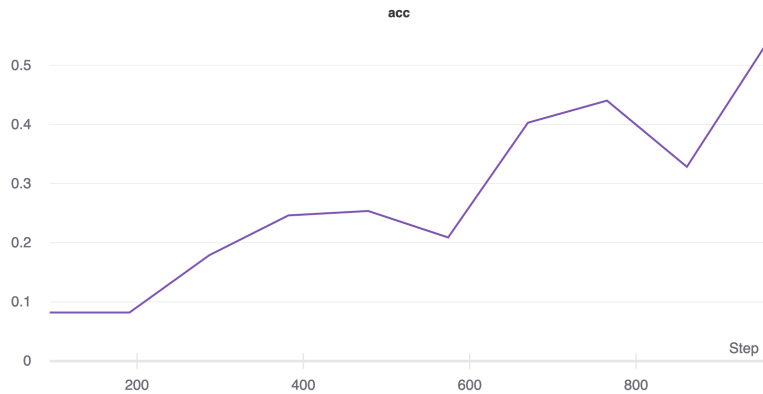


Figure 5.2: Accuracy of the first time running Yolov3

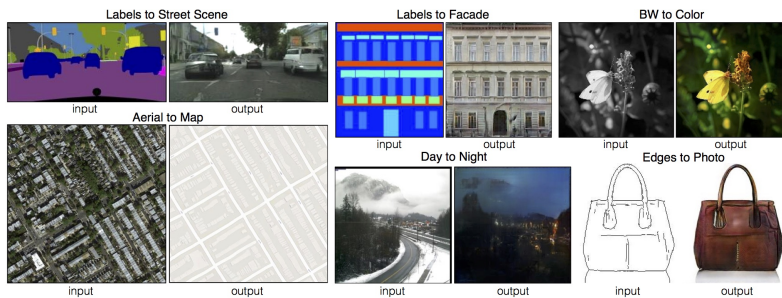


Figure 5.3: Example output of Pix2Pix, figure from [24]

Landry - H2O AI World London 2018” that they had gotten promising results with this model. So it might be worth investigating further by anyone willing to take a look.

Yolov5

Yolov5¹⁶ is the latest and best model built upon [25]. While we tested this model, we could see the loss going down, but this model did not seem to learn anything Figure 5.5.

We spent some time trying to debug why, trying to overfit on single examples and only one field, turning off augmentation, and turning off some of the weights with the pretrained model. None of it helped, and we decided to instead try a version with a smaller codebase and ended up with subsection 5.1.4.

5.2 The evolution of Yolov3

There were added some tweaks to the model to make it perform better. First of all, we applied a weighed dataloader that builds upon the process we discussed

¹⁶<https://github.com/ultraalytics/yolov5>

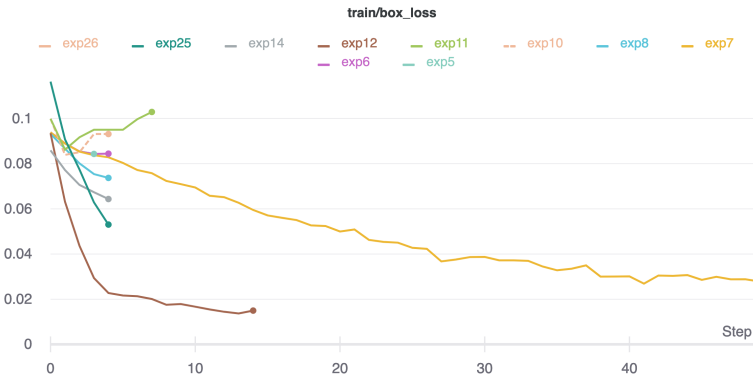


Figure 5.4: Loss of the bounding box for Yolov5

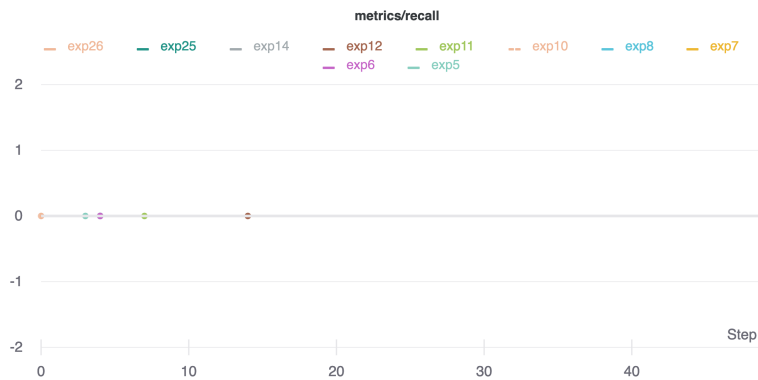


Figure 5.5: Flat recall curve for Yolov5

in subsection 4.4.2. We ran k-means on a invoice from each creditor, then based on the cluster we could see how much each invoice template contributed to the entire dataset. Then we could weight the dataset such that each invoice template is seen almost equally often, this reduced overfitting as can be seen in Figure 5.6.

We also imaug¹⁷ to run augmentation on the dataset. By applying Gaussian noise, shear, and rotation, we can get even more diverse data.

We also tried to fetch more of the more challenging samples, and uncommon invoices designs subsection 4.4.1, and they were weighted more in favor by the dataloader.

5.3 The application on top of the model

There needs to be some sort of extra software on top of the model to deal with input to the model, output of the model, cleaning up the model's output, running the GraphQL api, etc.

¹⁷<https://imgaug.readthedocs.io/en/latest/>



Figure 5.6: Average F1 score of all the fields. The first Yolov3 model vs the latest. Glamorous-shape-20 is name of the latest model.

This software is something we wrote our self and it makes no assumption about how the underlying model works. We expect only an input string. Therefore the model can be changed at any time.

This software is also what is in changer of validating the output of the model so it has built in account number and kid validation. As mentioned in subsection 2.1.1 some fields cannot be directly validated. So for instance the invoice number field and the customer number field we extract the number from the input.

5.4 Model output pipeline

The following describes what happens when an input is giving to the program.

5.4.1 Processing the image

First we preprocess the image. If it's a PDF it gets converted into an image. Currently only the first page is converted, but adding support for multiple pages should be quite straightforward.

After the image is converted it will be padded so that the image size is a square shape. This makes it a bit easier to work with the Yolo format.

5.4.2 Yolo bounding box prediction

Then we give the preprocessed image to the Yolo model to get the bounding box for each field. Yolo will then output a the bounding box location, class predictions and the confidence of each prediction.

5.4.3 Postprocessing of bounding box

Given an bounding box we will cut out the parts of the input image inside the bounding box.

```

1 from abc import ABC, abstractmethod
2 from typing import List
3
4
5 class TextResolver(ABC):
6     @abstractmethod
7     def find_in_text(self, text: str) -> List[str]:
8         """Resolves a field in a text"""
9
10    @abstractmethod
11    def find_in_file(self, path: str) -> List[str]:
12        """Resolves a field in a file"""

```

Figure 5.7: Resolver

5.4.4 EasyOCR for extracting the text

Based on the postprocessed bounding box image EasyOCR will extract the text.

5.4.5 Field revolvers

Borrowing a term from GraphQL¹⁸, field revolvers are classes we have made to be able to extract the field value from the OCR input and validate the value.

They all build upon the same abstract class shown in Figure 5.7 which makes it so we interact with all the resolves in the same way. The resolver are what decides how to extract the field value from an input string.

Some resolver will therefore be strict (like principal amount which will make the value invalid if it contains any non numeric value or missing a comma.

Other revolvers we don't need to be as strict because we can directly validate the value based on a validation schema Figure 5.8. Like the one for the account number. Here we preprocess the input a bit, then run the validate on the entire input and return the valid account numbers.

5.5 Testing

The main applications has unit tests for all the field revolvers. One can then easily add new rules for the resolver to make it stricter, and have tests to verify nothing broke.

It is not really feasible to run traditional unit tests on a machine learning model. That's what the validation set is for. We also used wandb a lot as mentioned in subsection 5.5.1. Tracking the model metrics made us sure we did not

¹⁸<https://graphql.org/learn/execution/>

```
1 from .modulus import Modulus
2 from .validator import Validator
3
4
5 class AccountNumberValidator(Validator):
6     def __init__(self) -> None:
7         self.modulus_10 = Modulus(10, [5, 4, 3, 2, 7, 6, 5, 4, 3,
8             ↪ 2])
9         self.modulus_11 = Modulus(11, [5, 4, 3, 2, 7, 6, 5, 4, 3,
10            ↪ 2])
11
12     def validate(self, input_str: str) -> bool:
13         account_number = input_str.strip()
14         has_correct_length = len(account_number) == 11
15         has_correct_check_digit =
16             ↪ self._has_correct_check_digit(account_number)
17         return has_correct_length and has_correct_check_digit
18
19     def _has_correct_check_digit(self, account_number: str) -> bool:
20         check_digit = int(account_number[-1])
21         account_number_without_check_digit = account_number[:-1]
22         calculated_check_digit_11 =
23             ↪ self.modulus_11.calculate_check_digit(account_number_without_check_digit)
24         return check_digit == calculated_check_digit_11
```

Figure 5.8: Account number validator



Figure 5.9: Example of the plots we generated with wandb

accidentally break the model. This could be further improved by having an additional dataset the model has to pass before becoming out in production.

5.5.1 Model health tracking

We also used Weights and Biases¹⁹ to track the model during training and to view historical performance against the newest model. For instance, we could this way detect that a bug had slipped in when we trained a new model because the F1 suddenly dropped more than expected as seen in Figure 5.10. We could also detect that we need to focus more on the date field, because it did not perform as well as the other fields Figure 5.9b.

5.6 Interacting with the model

We created a GraphQL API for interacting with the model. This way one can easily query the model. An example of the communication can be seen in Figure 5.11. This is for the file Figure 6.2.

The payment voucher model and the invoice model is trained separately. The reason for this is because of the difference in dataset size, and it makes it easier to debug things when we got unexpected results (because we know what data

¹⁹<https://wandb.ai/>

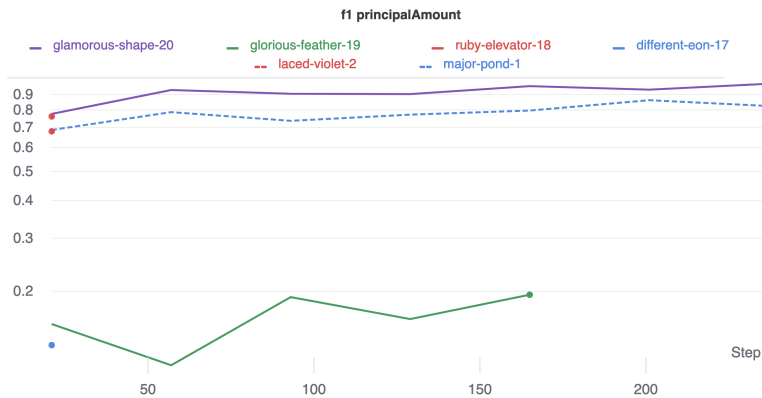


Figure 5.10: Sudden drop in F1, looks like a bug has slipped in...

can have affected the model). There is therefore two separate mutations “Upload-BankVoucher” and ‘ UploadInvoice”

5.7 The importance of a good frontend

The data outputted from the model still has to be validated. Even Tripletext does not automatically import the invoice it has parsed, so having a good frontend is crucial to make the application work nicely for the usecase. Google even has a option called “human in the loop”²⁰ where humans can correct the machine before the output is used. This is to prevent damage in critical business applications. Our plan is to use a good frontend so the user can validate in real time, and in a way such that any corrections can be used to further improve the model as prototyped in Figure 5.13. This way the frontend will both be a way to validate the model, and to automatically label any problems (which will be done any time the output value of the model is changed by the human). The human can correct the model by moving the bounding box for the field were the bounding box is off. Then the application will automatically read in the new value. This way the user does not need to type in anything, which can be error-prone and they also automatically label the dataset.

²⁰<https://cloud.google.com/document-ai/hitl>

```
mutation UploadInvoice($file
:Upload!) {
  UploadInvoice(file:$file)
  {
    bankDocument{
      invoiceNumber{
        value
      }
      invoiceAccountNumber{
        value
      }
      kid{
        value
      }
      date{
        value
      }
      dueDate{
        value
      }
    }
  }
}
```

Code listing (5.1) Request

```
{
  "data": {
    "UploadInvoice": {
      "bankDocument": [
        {
          "invoiceNumber": {
            "value": "37484"
          },
          "invoiceAccountNumber":
            {
              "value": "97753925280"
            },
          "kid": {
            "value":
              "0011616037484003"
          },
          "date": {
            "value": "2020-12-29"
          },
          "dueDate": {
            "value": null
          }
        }
      ]
    }
  }
}
```

Code listing (5.2) Response

Figure 5.11: Example of api request and response

FinanceDoc2Json



+ Select files

GraphQL

Figure 5.12: Our application frontend where the user can upload PDFs (Work in progress)

The screenshot displays a scanned invoice from Debet AS and a corresponding data entry form. The invoice details include:

- Sender:** Debet AS, Sagveien 23C, 0459 OSLO
- Customer:** Kjøper 10001, Fakturert: 08.01.2014
- Table:**

Varenr	Beskrivelse	Antall	Stk. pris	Mva	Rabatt	Netto
1	Konsulenttime	1	500,00	25%	0,00 (0%)	625,00
1	Konsulenttime betaling	3	900,00	25%	0,00 (0%)	2,700,00
-	Administrasjonsgodtyr	1	29,00	0%	0,00 (0%)	29,00
- Totals:** Nettoavgift: 3 229,00; Avg. pålegg: 3 200,00; Mva: 800,00; Rabatt: 0,00; **Totalt: 4 029,00**

The data entry form on the right contains the following fields:

- Kid: 034G
- Account number: 034G
- date: 034G
- due date: 034G
- invoice number: 034G
- customer number: 034G
- creditor number: 034G
- currency: 034G
- amount: 034G

A 'Lagre' button is located at the bottom of the form.

Figure 5.13: The page a user will be redirected to when having uploaded a document. The fields in the form to the right will be filled in by our machine learning model, through the GraphQL API. (Design prototype)

Chapter 6

Implementation and results

6.1 Invoices

As discussed in the subsection 4.4.2 we could cluster the invoice template into 35 cluster. If we ran the application on one from each cluster we got the following results Table 6.1. We will now show 5 examples of a input to the model, and the output so the numbers below makes more sense, and one can better understand what the model is able to do and not.

We also added the average edit distance, which is how "wrong" the output was. Low edit distance usually means error from the OCR. Th numerator is the total edit distance, and the denominator is how many files there was found an value, but has a mismatch with output.

The total accuracy was then 57.56%. 10% of the error is based on 3 chars edit errors of the output, which means a lot more accuracy can be gained with an better OCR model.

If one look at subsection 6.1.4 one can see that the $f - 1$ bounding box is much higher for some fields (like KID) and does not match with Table 6.1. Which can indicate bugs we have not found in both the post processing, and or the OCR.

Field	Correctly Classified	Missing/Incorrect	Edit distance
Kid	19	14	$\frac{44}{6}$
Date	20	15	$\frac{21}{7}$
Due Date	19	16	$\frac{23}{11}$
Principal amount	16	11	$\frac{25}{11}$
Customer number	20	15	$\frac{48}{14}$
Invoice account number	20	15	$\frac{2}{1}$
Invoice number	27	8	$\frac{32}{8}$

Table 6.1: Results on all the different invoice templates. Tested and validated by a human for all the 35 different templates.

6.1.1 Result of the latest model on a synthetic dataset

We created a synthetic dataset to preserve the privacy of the debtor and creditor of the original invoice. The template s based on a real invoice.

Example - 1

This is quite a basic invoice, and the template is quite common. As one can see from Figure 6.3 the model did all right. One can see in Table 6.2 the reason for the two misclassifications were not on our model, but from the OCR text extraction.

Field	Correct bounding box detected	Text extracted	Comment
Kid	✓	✓	
Date	✓	✓	
Due date	✓	✗	Got out invalid date. See Figure 6.1a
Principal amount	✓	✗	Read out decimal wrongly. See Figure 6.1b
Customer number	✓	✓	
Invoice account number	✓	✓	
Invoice number	✓	✓	

Table 6.2: Result matrix of Figure 6.2

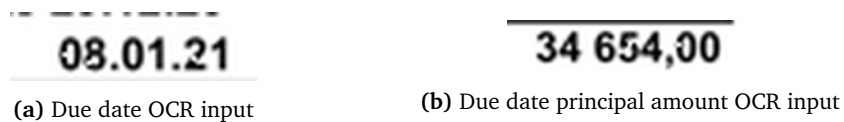



Figure 6.1: Error from example 1

	<p>SIGURD AS TORGGATA 72 2317 HAMAR</p> <p>Telefon 1881 E-post ntnu@stud.ntnu.no Web www.sigurdas.no Foretaksreg. NO 921229216 MVA</p>																																																																																																
<p>BRAGE AS TEKNOLOGIVEGEN 22 2815 GJØVIK</p> <p>Leveringsadresse TEKNOLOGIVEGEN 22 2815 GJØVIK</p>	<p>Faktura</p> <p>Faktura nr. 37484 Kundenr 11616 Fakturadato 29.12.20 Forfall 08.01.21 Ordre nr. 144561 Levert dato 19.03.20</p>																																																																																																
<p>FAKTURA GJELDER Installasjon anneks</p>																																																																																																	
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Beskrivelse</th> <th style="text-align: right;">Antall</th> <th style="text-align: right;">Enh</th> <th style="text-align: right;">Enh. pris</th> <th style="text-align: right;">Rabatt</th> <th style="text-align: right;">Beløp</th> </tr> </thead> <tbody> <tr> <td colspan="6">MEDGÅTT MATERIELL</td> </tr> <tr> <td>PR 2X2,5/2,5 SORT B-50</td> <td style="text-align: right;">50,0</td> <td>M</td> <td style="text-align: right;">36,29</td> <td style="text-align: right;">15,00%</td> <td style="text-align: right;">1 542,50</td> </tr> <tr> <td>TRIO GRAFITT LED 3K M/2XKON</td> <td style="text-align: right;">5,0</td> <td>STK</td> <td style="text-align: right;">1 563,26</td> <td style="text-align: right;">15,00%</td> <td style="text-align: right;">6 643,85</td> </tr> <tr> <td>LARGO GRAFITT 10W/830 IP65</td> <td style="text-align: right;">1,0</td> <td>STK</td> <td style="text-align: right;">1 431,83</td> <td style="text-align: right;">15,00%</td> <td style="text-align: right;">1 217,06</td> </tr> <tr> <td>ZIP SPOT KIT HVIT 3X6W/927 GU10</td> <td style="text-align: right;">1,0</td> <td>STK</td> <td style="text-align: right;">2 061,73</td> <td style="text-align: right;">15,00%</td> <td style="text-align: right;">1 752,47</td> </tr> <tr> <td>Fraktsone avgift</td> <td style="text-align: right;">1,0</td> <td>STK</td> <td style="text-align: right;">412,38</td> <td style="text-align: right;">15,00%</td> <td style="text-align: right;">350,52</td> </tr> <tr> <td>Fraktsone tillegg</td> <td style="text-align: right;">1,0</td> <td></td> <td style="text-align: right;">108,71</td> <td style="text-align: right;">15,00%</td> <td style="text-align: right;">92,40</td> </tr> <tr> <td>Fraktsone tillegg</td> <td style="text-align: right;">1,0</td> <td></td> <td style="text-align: right;">75,51</td> <td style="text-align: right;">15,00%</td> <td style="text-align: right;">64,18</td> </tr> <tr> <td colspan="6">MEDGÅTT ARBEID</td> </tr> <tr> <td>ARBEID - ORD. TID</td> <td style="text-align: right;">20,0</td> <td>TIM</td> <td style="text-align: right;">800,00</td> <td></td> <td style="text-align: right;">16 000,00</td> </tr> <tr> <td colspan="5">Totalt eks. mva</td> <td style="text-align: right;">27 663,00</td> </tr> <tr> <td colspan="5">Administrasjonsgebyr</td> <td style="text-align: right;">60,00</td> </tr> <tr> <td colspan="5">Øreavrunding</td> <td style="text-align: right;">0,20</td> </tr> <tr> <td colspan="5">Merverdiavgift 25.0%</td> <td style="text-align: right;">6 930,80</td> </tr> <tr> <td colspan="5">Totalt ink. mva</td> <td style="text-align: right;">34 654,00</td> </tr> </tbody> </table>		Beskrivelse	Antall	Enh	Enh. pris	Rabatt	Beløp	MEDGÅTT MATERIELL						PR 2X2,5/2,5 SORT B-50	50,0	M	36,29	15,00%	1 542,50	TRIO GRAFITT LED 3K M/2XKON	5,0	STK	1 563,26	15,00%	6 643,85	LARGO GRAFITT 10W/830 IP65	1,0	STK	1 431,83	15,00%	1 217,06	ZIP SPOT KIT HVIT 3X6W/927 GU10	1,0	STK	2 061,73	15,00%	1 752,47	Fraktsone avgift	1,0	STK	412,38	15,00%	350,52	Fraktsone tillegg	1,0		108,71	15,00%	92,40	Fraktsone tillegg	1,0		75,51	15,00%	64,18	MEDGÅTT ARBEID						ARBEID - ORD. TID	20,0	TIM	800,00		16 000,00	Totalt eks. mva					27 663,00	Administrasjonsgebyr					60,00	Øreavrunding					0,20	Merverdiavgift 25.0%					6 930,80	Totalt ink. mva					34 654,00
Beskrivelse	Antall	Enh	Enh. pris	Rabatt	Beløp																																																																																												
MEDGÅTT MATERIELL																																																																																																	
PR 2X2,5/2,5 SORT B-50	50,0	M	36,29	15,00%	1 542,50																																																																																												
TRIO GRAFITT LED 3K M/2XKON	5,0	STK	1 563,26	15,00%	6 643,85																																																																																												
LARGO GRAFITT 10W/830 IP65	1,0	STK	1 431,83	15,00%	1 217,06																																																																																												
ZIP SPOT KIT HVIT 3X6W/927 GU10	1,0	STK	2 061,73	15,00%	1 752,47																																																																																												
Fraktsone avgift	1,0	STK	412,38	15,00%	350,52																																																																																												
Fraktsone tillegg	1,0		108,71	15,00%	92,40																																																																																												
Fraktsone tillegg	1,0		75,51	15,00%	64,18																																																																																												
MEDGÅTT ARBEID																																																																																																	
ARBEID - ORD. TID	20,0	TIM	800,00		16 000,00																																																																																												
Totalt eks. mva					27 663,00																																																																																												
Administrasjonsgebyr					60,00																																																																																												
Øreavrunding					0,20																																																																																												
Merverdiavgift 25.0%					6 930,80																																																																																												
Totalt ink. mva					34 654,00																																																																																												
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="4" style="text-align: left;">Betalingsinformasjon</th> </tr> </thead> <tbody> <tr> <td style="width: 33%;">Kundenr 11616</td> <td style="width: 33%;">Faktura nr. 37484</td> <td style="width: 33%;">Kontonr 9775.39.25280</td> <td style="width: 33%;"></td> </tr> <tr> <td>SIGURD AS</td> <td>Fakturadato 29.12.20</td> <td>KID nr 0011616037484003</td> <td></td> </tr> <tr> <td></td> <td>Forfall 08.01.21</td> <td>Å betale 34654,00</td> <td></td> </tr> </tbody> </table>		Betalingsinformasjon				Kundenr 11616	Faktura nr. 37484	Kontonr 9775.39.25280		SIGURD AS	Fakturadato 29.12.20	KID nr 0011616037484003			Forfall 08.01.21	Å betale 34654,00																																																																																	
Betalingsinformasjon																																																																																																	
Kundenr 11616	Faktura nr. 37484	Kontonr 9775.39.25280																																																																																															
SIGURD AS	Fakturadato 29.12.20	KID nr 0011616037484003																																																																																															
	Forfall 08.01.21	Å betale 34654,00																																																																																															

(a) Input

```

1  [
2  {
3    "kid": "0011616037484003",
4    "date": "2020-12-29",
5    "due_date": null,
6    "principal_amount": "34 654,90",
7    "customer_number": "11616",
8    "invoice_account_number": "97753925280",
9    "invoice_number": "37484"
10 }
11 ]

```

(b) Output

Figure 6.2: Results of example 1

Example - 2

This is a bit more complicated invoice. As mentioned in the subsection 4.4.2 this invoice design does not come up very often. Even many of the existing solutions had problems with this one as discussed in chapter 7, as did our solution Figure 6.3. One can see the results in Table 6.3 and Figure 6.4.

Field	Correct bounding box detected	Text extracted	Comment
Kid	X	X	It's the account number
Date	X	X	
Due date	X	X	It's the date
Principal amount	✓	X	EasyOCR added a backslash as part of the output which cut part of the input off
Customer number	X	X	This is actually misread organization number
Invoice account number	X	X	The bounding box is bit off, but almost well placed.
Invoice number	X	X	

Table 6.3: Result matrix of Figure 6.4

17 910,00

(a) Principal amount OCR input

DAIKKOITO.

97753546107

(b) Invoice account number OCR input

Figure 6.3: Error from example 2

Opr./Artikel-nummer	Beskrivelser	Antal	Pris	Rab	Summa
A	ARBEID FOTWELL MODUL MED PROG				1 520,00
A	ARBEID DEMPERS BAK				950,00
A	BREMSE BYTTE BAK MED SKIVER OG KALIPPER				1 710,00
D	MODUL FOTWELL	1,00	4 274,00	10%	3 846,60
D	KLOSSER BAK	1,00	904,00	20%	723,20
D	SKIVER BAK	2,00	940,00	20%	1 504,00
D	KALIPER BAK	1,00	1 560,00	20%	1 248,00
D	DEMPERS BAK	2,00	1 531,00	20%	2 449,60

Kunde mottaker SIGURD AS Bokart: 2020-01-13 0:00		Skadenummer	Deres ref:	ORG.No/MVA 921229216
Eieren: SIGURD.NO SIGURD AS	Tel. mobil	Driver: KONTANTKUND	Tel. mobil	
Reg.nr:	Merke:	Modell:	Årsmodell: 2006	Første reg.dat: 2007-10-19
			Kilometer:	Chassinummer:

«Oppfølging av alle våre fakturaer er overlagt til Finexa. Ved for sen betaling påløper 2 % forsinkelsesrente pr mnd mot næringsdrivende og lovregulert forsinkelsesrente mot private, samt purreomkostninger.»

Moms 25,00% 3581,90	Sammendrag
Merknader ikke kommet innen 8 dager er ikke godkjent.	Arbeids exkl. 4 180,00
Etter forfallsdato belastes renter	Deler exkl. 9 771,40
	Forbruksvarer ex.MVA. 376,20
	Sammendrag ex. MVA 14 327,60
	MVA vil 3 581,90
	Øreavrundning 0,50
	Å betale 17 910,00

SIGURDS BILVERKSTED AS TORGGATA 72 2317 HAMAR	ORG.NR / NO 921229216 MVA Mail: bragearn@stud.ntnu.no Hjemmeside: Registrert i Foretaksregisteret	Tel: Bankkonto: 97753546107 Side 1 (1)
---	--	---

(a) Input

```

1 {
2   "invoiceNumber": "17",
3   "invoiceAccountNumber": null,
4   "kid": "97753546107",
5   "dueDate": "2020-01-13",
6   "date": null,
7   "customerNumber": "991990946",
8   "principalAmount": "910,00"
9 }

```

(b) Output

Figure 6.4: Results of example 2

Example - 3

This invoice is also a bit complicated since the customer number is not very easy to see. The model has also problems finding that, and the due date, but does otherwise well. In both cases the problem is our model does not find the correct bounding box. Figure 6.5

Field	Correct bounding box detected	Text extracted	Comment
Kid	✓	✓	
Date	✗	✗	
Due date	✓	✓	
Principal amount	✓	✓	
Customer number	✗	✗	
Invoice account number	✓	✓	
Invoice number	✓	✓	

Table 6.4: Result matrix of Figure 6.5

BRAGE AS TORGGATA 72		Telefon: Fak.: E-post: brageam@stud.ntnu.no Org.nr: 921229216 Foretaksregister: 921229216 MVA	Faktnr.: 152150 Fakt.dat: 06.05.19 Forf.dat: 05.06.19 KID: 00011521509 Kontonr.: 9775.47.75878	Side: 1/1			
2317	HAMA						
SIGURD AS Teknologivegen 22		Ordrenr.: 159681	Leveringsadresse: Sigurd Teknologivegen 22				
2815	GJØVIK	Ordredato: 04.05.19	2815	GJØVIK			
Kundnr.:	31694	Vår ref.: att					
		Deres ref.:					
		K. best.nr.:					
		Fors.type: Fraktfritt: Ja					
Faktura							
Artikkelnr.	Artikkelnavn	Enh.pris	Rab.I	Rab.II	Lever	Rest	Beløp
WebOrderID: 742048							
4230183	BOX008 RETTANGOLO Krom Servantbatteri 234 mm	5 036,00	38,00	5,00	3,00		8 898,61
Sum ordrelinjer:		8898,61	Fakturagebyr:		Sum mva:		2224,65
Fakturabatt:			Porto:		Fakturabeløp:		11 123,26
Kontantrabatt:			Mva grunnlag:		8898,61		

Vikem Software AS - Vikem Global (41118819)

(a) Input

```

1 {
2   "invoiceNumber": "152150",
3   "invoiceAccountNumber": "97754775878",
4   "kid": "00011521509",
5   "dueDate": "2019-05-06",
6   "date": null,
7   "customerNumber": null,
8   "principalAmount": "11 123,26"
9 }

```

(b) Output

Figure 6.5: Results of example 3

6.1.2 Result of the latest model on a out of sample dataset

Since we all have some invoices laying around, we decided to test the model on some invoice templates it had never seen before.

NTNU invoice

Field	Correct bounding box detected	Text extracted	Comment
Kid	✓	✓	
Date	✗	✗	Thought is was the term field. Therby failing validation
Due date	✗	✗	Is the date.
Principal amount	✓	✓	
Customer number	✓	✓	There was no customer number here. So "null" is correct.
Invoice account number	✓	✓	
Invoice number	✓	✗	Misclassified a 3 as 9. Position 5.

Table 6.5: Result matrix of Figure 6.6

Faktura

Sigurd Santiago Schoeler



Faktura 1106384
 Termin VÅR 2021
 Fakturadato 29.04.2021
 Betalingsfrist
 KID 521342011063845
 Kontonr. 7694 05 17431
 Utestående 0,00
 MVA 0,00
 Referanse
 Fakturaref

Som student må du betale semesteravgift og registrere deg hvert semester. Dette kan du lese mer om på Innsida. Hvis du har aktiv studierett, har fullført semesterregistreringen og innbetalingen er registrert, oppdateres studentbevis-appen din automatisk og Lånekassen får bekrefteelse om at du er aktiv student. Semesteravgiften går tilbake til studentene ved å finansiere velferdstilbud. Vil du vite mer om hva semesteravgiften går til, se www.velferdstinget.no.

Husk å legge inn riktig KID-nummer og sjekk gjerne at fakturaen gjelder riktig semester. Merk at SAIH-tierne er valgfritt å betale, og om du ikke ønsker å støtte SAIH må du trekke beløpet merket SAIH fra totalbeløpet. SAIH-tierne går til Studentenes og akademikernes internasjonale hjelpefond (SAIH). Vil du vite mer om hva du støtter med SAIH-tierne, se www.saih.no.

Beskrivelse	Beløp
Semesteravgift til Studentsamskipnaden i Gjøvik, Ålesund og Trondheim	590,00
Valgfri støtte til Studentenes og akademikernes int. hjelpefond	40,00
OCR (innlest 26.01.2021)	-580,00
Frafall for valgare beløp.	-40,00
Totall utestående beløp (NOK)	0,00

Norges teknisk-naturvitenskapelige universitet
 7491 TRONDHEIM

Org.nr. NO 974767880 MVA
 Kontonr. 7694 05 17431
 SWIFT DNBANOKXXXX
 IBAN NO1576940517431

(a) Input

```

1 {
2   "kid": "521342011063845",
3   "date": null,
4   "due_date": "2021-04-29",
5   "principal_amount": "0,00",
6   "customer_number": null,
7   "invoice_account_number": "76940517431",
8   "invoice_number": "1106984"
9 }
```

(b) Output

Figure 6.6: Results of NTNU invoice

Telenor Invoice

This is also an invoice design the model has never seen before. It is also quite unique and does not look like most of the invoices in the dataset. In addition to that is this a scanned input image. The model is actually able to find the most obvious fields, but misclassified the customer number, and did not find the account number.

Field	Correct bounding box detected	Text extracted	Comment
Kid	✓	✓	
Date	✗	✗	It's below the customer number
Due date	✓	✓	
Principal amount	✓	✓	
Customer number	✗	✗	
Invoice account number	✗	✗	
Invoice number	✗	✗	Not as apparent as other invoices, but it is present at the top right.

Table 6.6: Result matrix of Figure 6.7

telenor

Retur: Telenor Norge AS, Strandgata 9, 7900 RØRVIK

EA

BRAGE ARNKVÆRN

Kundeservice Side 1 av 2

telenor.no/faktura

9 15 09000

Kundenummer/eFakturareferanse: 5511972134
 Ansvarlig: BRAGE ARNKVÆRN
 Telefonnummer / Bruker: BRAGE ARNKVÆRN

BRAGE ARNKVÆRN
 Fakturanummer: 5511972134009
 Fakturadato: 13.04.2021

Faktura

Beløp å betale: 2.131,86 NOK

Betalingsfrist: 27.04.2021

Informasjon

Med Mitt Telenor-appen kan du:
 * Se, betale eller utsette fakturaen din
 * Få full oversikt over mobilforbruket
 * Fylle på data
 * Endre abonnement
 Last ned appen ved å sende MT til 1999

Ønsker du å unngå gebyr for papirfaktura og få fakturaen direkte i nettbanken, kan du bestille eFaktura neste gang du besøker nettbanken.
 eFakturareferansen finner du oppe til høyre på fakturaen.

Kort fortalt

Engangspriser	1.576,00
Faste priser	549,00
Bruk	6,86
Totalt	2.131,86

Se baksiden for detaljer:

	MVA	Grunnlag NOK	MVA NOK
Telenor Norge AS	0%	17,85	0,00
Kundeservice Privat	25%	1.691,21	422,80
Postboks 216 Sentrum	SUM	1.709,06	422,80
3701 SKIEN			

Fra utlandet: +47 9 15 09000

Foretaksregisteret: NO 976 967 631 MVA
 SWIFT/BIC: NDEANDKK
 IBAN: NO21 6005 0777 122

Kvittering	Innbetalt til	Beløp	Betalersens kontonummer	Blåketnummer
RIV HER!	6005 07 77122	2.131,86		<6851684095>

GIRO Betalingsfrist: 27.04.2021

Underskrift ved girering

Betalt av: BRAGE ARNKVÆRN

Betalt til: Telenor Norge AS, Mobil, Strandgata 9, 7900 RØRVIK

Belast konto:

Kvittering tilbake:

Kundidentifikasjon (KID)	Kroner	Øre	Til konto	Blåketnummer
H 55119721340091	2131	86	< 0 > 6005 07 77122	<6851684095>

(a) Input

```

1 {
2   "invoiceNumber": null,
3   "invoiceAccountNumber": null,
4   "kid": "55119721340091",
5   "dueDate": "2021-04-27",
6   "date": null,
7   "customerNumber": "27",
8   "principalAmount": "2131,86"
9 }

```

(b) Output

Figure 6.7: Results of the Telenor invoice

Komplett invoice

The following is technically an receipts, but has all the fields an invoice would have. So we decided to test it. The model does not as well here.

Field	Correct bounding box detected	Text extracted	Comment
Kid	X	X	
Date	X	X	Due date and date is the same, but here the model does not output a value for the date, but it is defined on the invoice.
Due date	✓	✓	
Principal amount	✓	✓	
Customer number	X	X	
Invoice account number	X	X	
Invoice number	X	X	

Table 6.7: Result matrix of Figure 6.8



Kvittering

Leveringsadresse Brage Arnkværn Torggata 72 N-2317 Hamar	Bestiller Brage Arnkværn Torggata 72 N-2317 Hamar	Faktura nr 914523425	
		Faktura dato 28.12.2015	
		Forfallsdato 28.12.2015	
		Side 1 av 1	
		Kunde nr 8634865	
Fakturamottaker Brage Arnkværn Torggata 72 N-2317 HAMAR	Transportør PostNord	Ordre nr 27162824	
	Kid nr 8153570018	Ordre dato 28.12.2015	
	Bestiller Brage Arnkværn	Lev.bekr.no 815357001	
	Vår ref WEB	Lev.dato 28.12.2015	
	Din ref	Betaling Visa	
		Valuta NOK	
Linje	Produkt/Beskrivelse	Antall	Enhetspris
100	757761 WD Red 1TB NAS Harddrive SATA 6Gb/s (SATA 3.0), 64MB, 3.5", 24x7 reliability, IntelliPower, NASware	4 STK	559,20
			Sum
			2.236,80
		Netto	2.236,80
		Frakt	0,00
		Mva 25%	559,20
		Fakturasum	2.796,00
Betaling gjort via Visa	2.796,00		

Denne faktura er belastet kortet som ble oppgitt ved bestilling, og skal derfor ikke betales manuelt.

KID nr.	Belep	Kontonummer
8153570018	2.796,00	9750.07.06922

Pakkesporing:

Ønsker du å returnere en vare?
 Dette er enkelt å gjennomføre ved å søke retur på www.komplett.no/retur.
 Har du andre spørsmål rundt ditt produkt kan Komplett Kundesenter veilede deg på telefon (33 00 55 00) eller mail (salg@komplett.no, support@komplett.no).

Har du handlet privat og ønsker angrefrist?
 Fremgangsmåte for å søke angrefrist finner du her: www.komplett.no/angrefrist.

Komplett.no Komplett Services AS Postboks 2094 NO-3202 Sandefjord	Tlf. Faks E-post Internett	33005500 33005001 kundeservice@komplett.no www.komplett.no	Foretaksnr. NO 979642121 MVA Bankkonto 9750.07.06922 Foretaksregisteret
--	-------------------------------------	--	---

(a) Input

```

1 {
2   "invoiceNumber": "28",
3   "invoiceAccountNumber": null,
4   "kid": null,
5   "dueDate": "2015-12-28",
6   "date": null,
7   "customerNumber": "07",
8   "principalAmount": "2796,00"
9 }
```

(b) Output

Figure 6.8: Results of the Komplett invoice

6.1.3 How much augmentation can the model handle ?

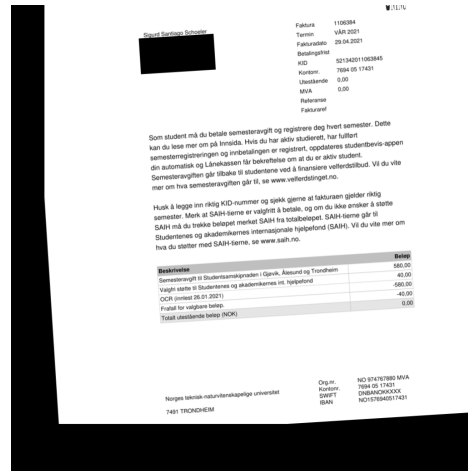
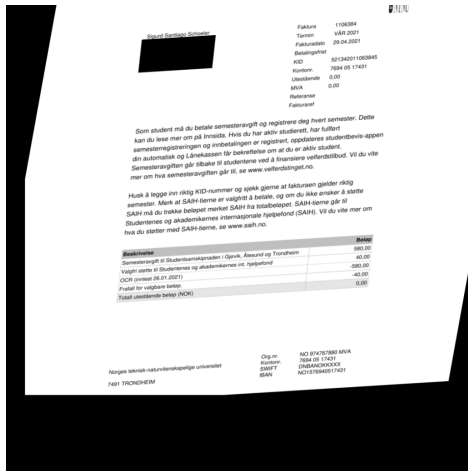
Because we know the input to the model could become quite noisy because of the input coming from scanners, or bad photo images. It's worth checking out how much augmentation the model can handle. By applying some augmentation Figure 6.9 on a image not in the training set and giving it to the model we got the following results.

Kid and due date stayed the same through all the augmentation. Which was the same results as in Figure 6.6.

Customer number was misclassified as '0' when the rotation was applied, but was correct with the other augmentations.

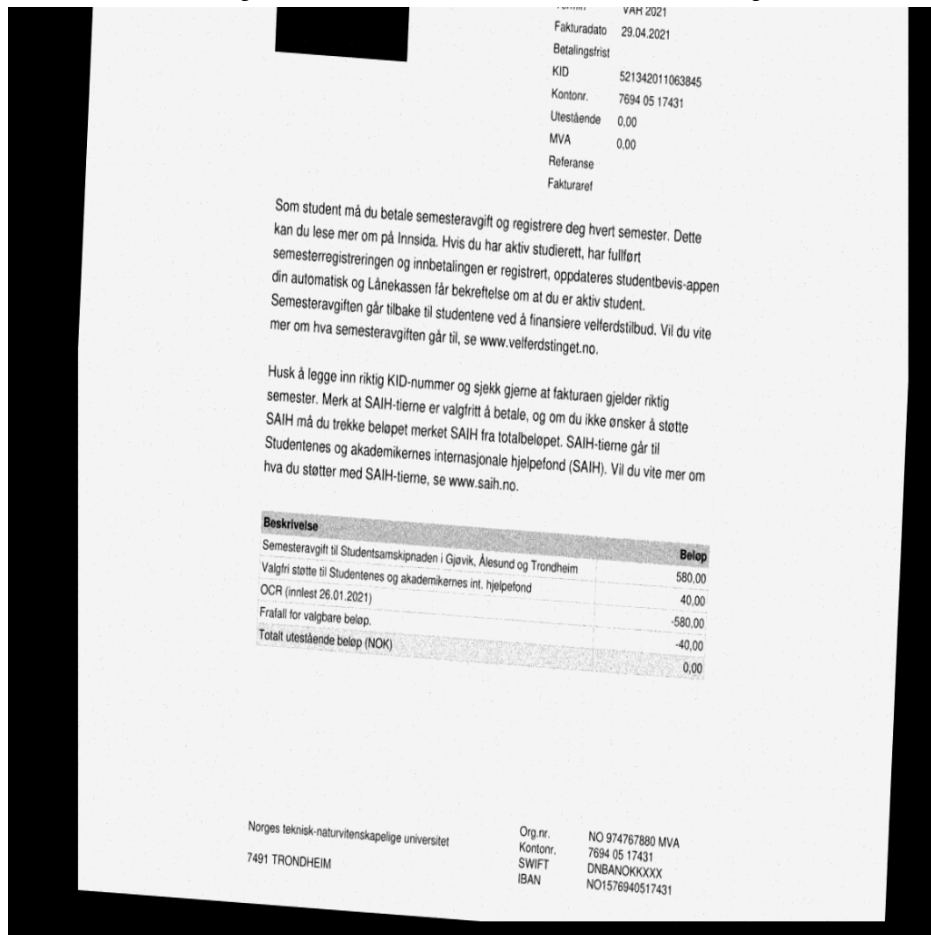
Principal amount was correct with rotation, but was set to "480" when shearing was applied, and failed validation when the scanning augmentation was applied as the OCR returned "U,000c".

Invoice number was correct with shear and rotation, but not the scanning augmentation. Date was null on all, like we got with Figure 6.6.



(a) Shear augmentation

(b) Rotation augmentation



(c) Scanning augmentation (rotation, shear, blurring and gaussian noise)

Figure 6.9: Augmentation

6.1.4 F1-score of our model on bounding box matching on invoices

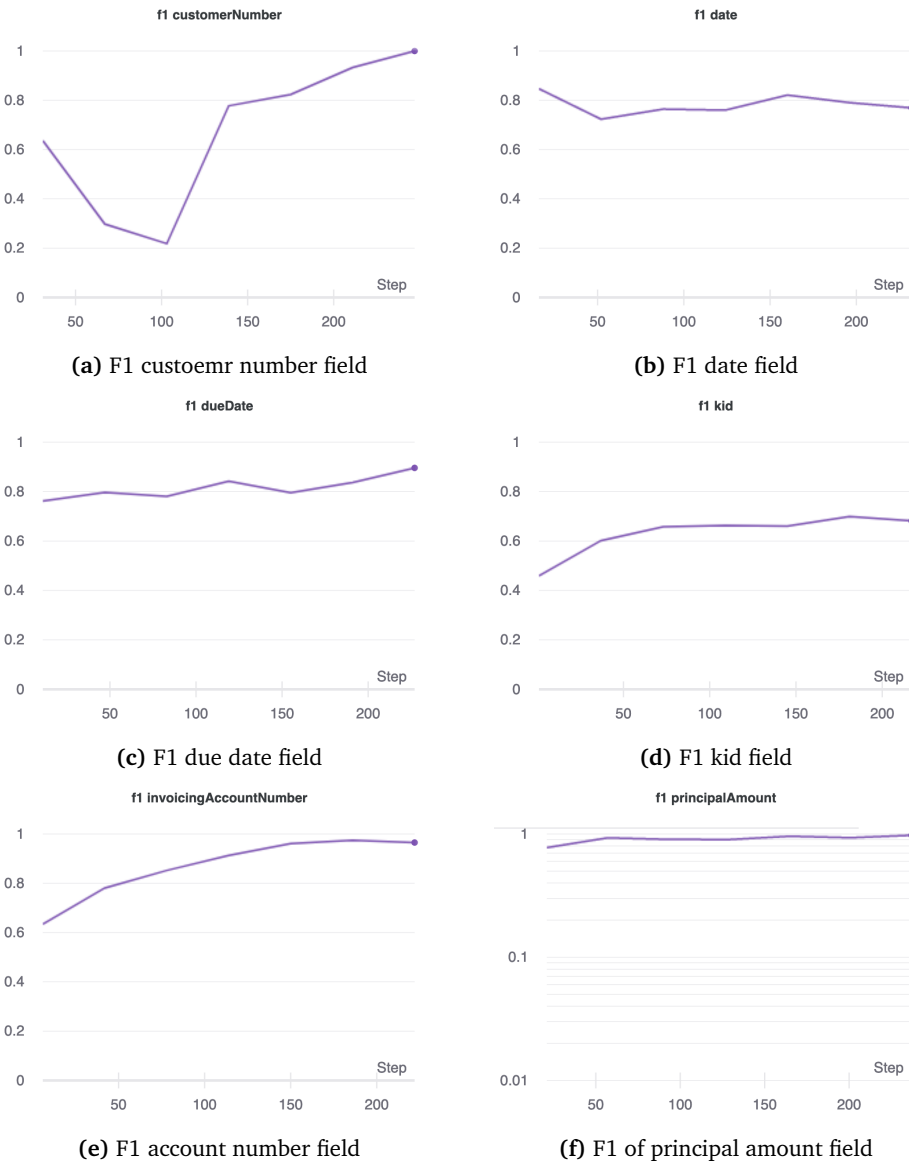


Figure 6.10: wandb logs from our last invoice model

6.2 Payment vouchers

Since payment vouchers will always come directly from the bank we can here assume the quality of the input to be much higher. We did however also here apply things like augmentation to try to achieve the best quality. We can however see from Figure 6.15 that there are cases the application has problems with.

6.3 Result of the latest model on a synthetic dataset

6.3.1 Example 1

This payment voucher was parsed without any problems. One can however see that the model gives a bounding box that is a bit big, so one could add some more post processing for the fields message, payer and creditor. However the output is correct.

```

1  [
2    {
3      "fromAccountNumber": "97753925280",
4      "toAccountNumber": "97754775878",
5      "message": "BELOPET GJEL0ER:\nF3122931/KN3124766\n\f",
6      "creditor": "BETALER:\nFinexa AS\nTorggata 72\n2317 Hamar\n\f",
7      "payer": "BErALEen,\nFinexa AS\nTorggata 72\n2317 Hamar\n\f",
8      "principalAmount": "3418,00",
9      "date": "2020-09-25"
10   },
11   {
12     "fromAccountNumber": "97753546107",
13     "toAccountNumber": "97754775878",
14     "message": "BELOPET GJEL0ER:\nKundenr. 18534\n\f",
15     "creditor": "\f",
16     "payer": "BETALER:\n\nNTNU GJOVIK\nTEKNOLOGIVEGEN 22\n2815
17     ↪ Gjovik\n\f",
18     "principalAmount": "1749,00",
19     "date": "2020-09-25"
20   }
21 ]

```

(a) Output

Figure 6.11: Results of basic/medium payment voucher

Denne forsendelsen inneholder meldinger for 25. september 2020.
Eventuelle spørsmål kan rettes til Nordea på telefon 232 06001 (privatkunder) eller 232 06002 (bedriftskunder).

Nordea

SIGURD OG BRAGE SITT BAKKERI AS
Torggata 72
2317 HAMAR

**Forsendelsen inneholder 2 meldinger for
Konto 9775.47.75878:**
2 meldinger om kreditering

E-post: brage@stud.ntnu.no

ENKELTBELØP ført på konto

MELDING OM KREDITERING	
BETALER: Finexa AS Torggata 72 2317 Hamar	MOTTAKER: SIGURD OG BRAGE SITT BAKKERI AS
BELOPET GJELDER: F3122931/RN3124766	

Oppgj. dato : 25/09-2020
Oppdr. dato : 25/09-2020
Til konto : 9775.47.75878
Fra konto : 9775.39.25280
Beløp : 3.416,00
Blankettnr. : 338283233
Arkivref. : 32838233
ID nummer : 3827737233

MELDING OM KREDITERING	
BETALER: NTNU GJØVIK TEKNOLOGIVEGEN 22 2815 Gjøvik	MOTTAKER:
BELOPET GJELDER: Kundenr. 18534	

Oppgj. dato : 25/09-2020
Oppdr. dato : 24/09-2020
Til konto : 9775.47.75878
Fra konto : 9775.35.46107
Beløp : 1.749,00
Blankettnr. : 128432346
Arkivref. : 323287371
ID nummer : 0000000000

6.3.2 Example 2

This payment voucher was parsed without any problems.

```
1 {
2   "data": {
3     "UploadBankVoucher": {
4       "bankDocument": [
5         {
6           "fromAccountNumber": {
7             "value": "97753925280"
8           },
9           "toAccountNumber": {
10            "value": "97754775878"
11          },
12          "message": null,
13          "creditor": {
14            "value": "SIGURD AS\n\f"
15          },
16          "payer": {
17            "value": "BRAGE\nteKnologivegen 22\n\f"
18          },
19          "principalAmount": {
20            "value": "217,24"
21          },
22          "date": {
23            "value": "2020-09-23"
24          }
25        }
26      ]
27    }
28  }
29 }
```

(a) Output

Figure 6.13: Results of basic payment voucher



Retur: SpareBank 1 SR-Bank ASA,
Postboks 250, 4068 STAVANGER

Dato 23.09.2020

Sidenr. 1

3229

Org.nr. NO 937895321 Foretaksregisteret

Sigurd AS
Epost:
bragearn@sutd.ntnu.no

Innbetalingsoversikt for konto: 9775.47.75878

Bokført dato:	Beløp:	Betalar:	Mottakar:	Referanse:
23.09.2020	217,24	BRAGE TEKNOLOGIVEGEN 22	SIGURD AS	
Valuta dato:	23.09.2020			237327878110
Fra konto:	9775.39.25280			

(a) Input

6.3.3 Example 3

This payment voucher with multiple entries gave an unexpected results as can be seen Figure 6.15. The reason the JSON response is so weird, seems to not be the model, but the software on top. We did not have time to debug this, but add it to show the software has some problems with parsing payment vouchers as well.

```
1  [
2    {
3      "fromAccountNumber": {
4        "value": "97753925280"
5      },
6      "toAccountNumber": null,
7      "message": null,
8      "creditor": {
9        "value": "BRAGEITAS\ntEKNOLOGIVEGEN 22\n2815 GIOVIK\n\f"
10     },
11     "payer": {
12       "value": "NTNU GIOVIK\ntEKNOLOGIVEGEN 22\n2815 GJOVIK\n\f"
13     },
14     "principalAmount": {
15       "value": "71484,40"
16     },
17     "date": {
18       "value": "2020-09-16"
19     }
20   },
21   {
22     "message": {
23       "value": "iaiciiihedamabl\n\f"
24     },
25     "creditor": {
26       "value": "BRAGE IT AS\n\f"
27     },
28     "payer": null,
29     "principalAmount": null,
30     "date": {
31       "value": "2020-09-16"
32     }
33   },
34   ...
```

(a) Output part 1

```
1     ...
2     {
3         "fromAccountNumber": {
4             "value": "97753546107"
5         },
6         "toAccountNumber": null,
7         "message": null,
8         "creditor": {
9             "value": "| BRAGEIT AS\nTEKNOLOGIVVEGEN 22\n2815 GIOVIK\n\f"
10        },
11        "payer": {
12            "value": "7 SIUURY UU BRAVE SIDE DARREN\n0 0125 OSLO\n7\n\f"
13        },
14        "principalAmount": {
15            "value": "150000,00"
16        },
17        "date": {
18            "value": "2020-09-16"
19        }
20    },
21    {
22        "fromAccountNumber": {
23            "value": "97753925280"
24        },
25        "toAccountNumber": null,
26        "message": null,
27        "creditor": {
28            "value": "BRAGE IT AS\n\f"
29        },
30        "payer": {
31            "value": "/ BRAGE FINANS\nPOSTBOKS 100\n| 2317 HAMAR\n\f"
32        },
33        "principalAmount": {
34            "value": "150000,00"
35        },
36        "date": {
37            "value": "2020-09-16"
38        }
39    }
40 ]
```

(b) Output part2

Figure 6.15: Results of hard payment voucher



Retur:
SpareBank 1 SMN, 7467 TRONDHEIM

4210

Brage IT AS
Epost:
bragearn@stud.ntnu.no

Dato 16.09.2020

Sidenr. 1

Org.nr. NO 937901003 Foretaksregisteret

Privat: 915 07300/ smn@smn.no
Bedrift: 915 07303/ smn.bedrift@smn.no
Hjemmeside: smn.no

Innbetalingsoversikt for konto: 9775.47.75878

Bokført dato:	Beløp:	Betaler:	Mottaker:	Referanse:
16.09.2020	4.943,93	SIGURD AS	BRAGE IT AS	
Valuta dato:	16.09.2020			43243243243243243243
Fra konto:	9775.47.25280			
Beløpet gjelder:				
BETALING AV 288477				
16.09.2020	71.484,40	NTNU GJØVIK TEKNOLOGIVEGEN 22 2815 GJØVIK	BRAGE IT A S TEKNOLOGIVEGEN 22 2815 GJØVIK	324324324324324324234
Valuta dato:	16.09.2020			
Fra konto:	9775.39.25280			
Beløpet gjelder:				
WB034959				
16.09.2020	140.000,00	SIGURD OG BRAGE SITT BAKKERI	BRAGE IT AS TEKNOLOGIVEGEN 22	
Valuta dato:	16.09.2020	0125 OSLO	2815 GJØVIK	3432234325324324324
Fra konto:	9775.35.46107			
Beløpet gjelder:				
WB72847				
16.09.2020	150.000,00	BRAGE FINANS POSTBOKS 100	BRAGE IT AS	
Valuta dato:	16.09.2020	2317 HAMAR		32432423432532532
Fra konto:	9775.39.25280			
Beløpet gjelder:				
210915		150000,00	150.000,00	
16.09.2020	259.900,00	SIGURD AS TORGGATA 72	BRAGE IT A/S	
Valuta dato:	16.09.2020	2317 HAMAR		4324323432344323
Fra konto:	9775.47.25280			
Beløpet gjelder:				
PP423273				

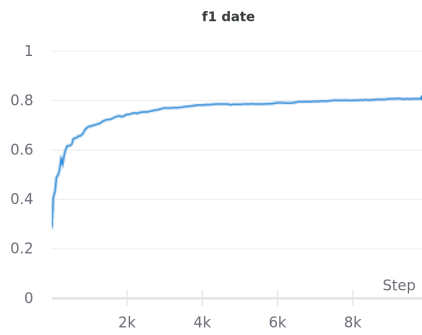
Fortsetter på neste side

(a) Input

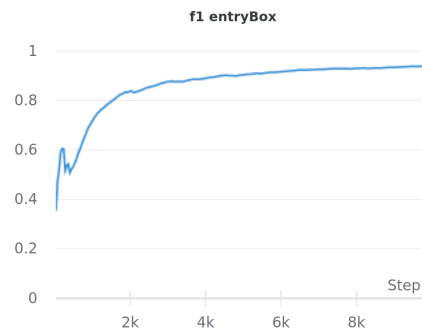
6.3.4 F1-score of our model on payment vouchers

We cannot do the same quantitative analysis of the accuracy like we did section 6.1, because we do not have the same JSON file.

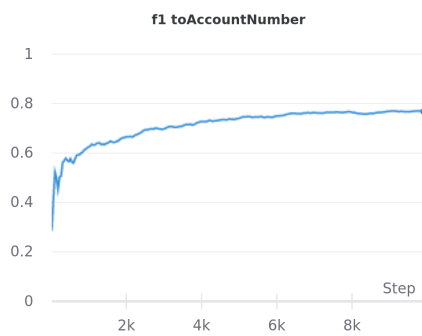
Because this dataset is so small, we applied smoothing to f1-score and got the following results. As one can see the model is on the right direction. This f1-score is based on the bounding box, and not the final output as mentioned above. This can be seen in Figure 6.17.



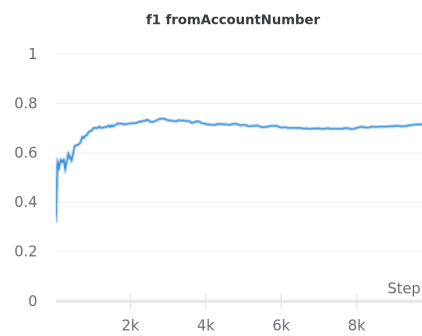
(a) F1 smoothed to date field



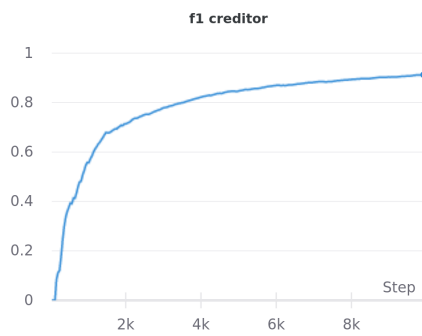
(b) F1 smoothed to entry box field



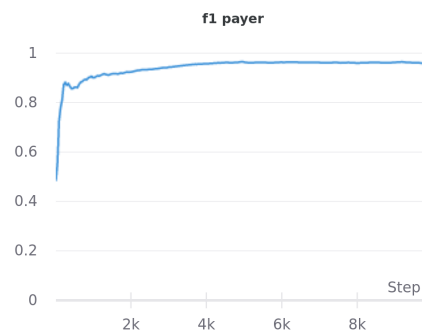
(c) F1 smoothed to account number field



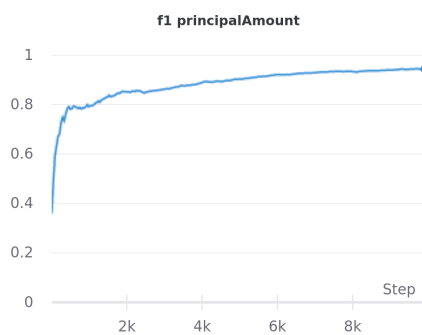
(d) F1 smoothed from account number field



(e) F1 smoothed creditor field



(f) F1 smoothed payer field



(g) F1 smoothed principal amount field

Figure 6.17: wandb results from the latest bankvoucher model

Chapter 7

Overview of existing solutions

There already some existing solution, they are however not usually tailored towards getting a clean JSON-like file out from the system, or they are paid or closed source. Common with all these solutions is that they did not correctly identify the date in the invoice in Figure 6.4, which tells us that this is an uncommon invoice layout.

7.1 DocParser

DocParser is an online solution to parse and extract info from PDF's. A user can upload PDF's and choose from a selection of already available templates to extract information. DocParser has different rules for extracting text. The rules work by either specifying the location of the field inside the document and extract the text in this position, or by converting the document to text and using simple text editing and conversion rules to locate a value. Pros: simple solution that non-developers can use to extract fields. cons: the rules have to be specified manually, and we would have to create a ruleset for every different invoice layout. This does not scale well, because most people receive invoices with different formats and layouts. The solution can easily be recreated in a programming environment using regex, and does not need machine learning. This might be viable to use when parsing bank vouchers, as there are only a small amount of layouts, and it is viable to create a custom document parser for each of these layouts.

7.2 TripleTex

TripleTex is a Norwegian accounting system, which has a module for analyzing invoices. This solution allows for extracting relevant fields from invoices. It is however closed sourced and only works on invoices, and not other financial document types like Payment Vouchers or receipts.

7.2.1 Accuracy

We have tested TripleTex' fakturatolk manually with 17 different invoice layouts, where every invoice infers 5 different fields which means we read 85 different values. Out of these 85 values, 4 of them were present on the invoice but could not be found, and there were 3 fields were fakturatolk suggested false values, which means their accuracy is 91.7%. Fakturatolk did however also have challenges with parsing Figure 6.4, and did not parse that correctly.

Tripletex does not read the invoice lines, does not set the supplier if it does not already exists in the database, and it does not read the KID and account number.

7.3 Nanonets

Nanonets is a service which provides computer vision models, to extract key information. The service is commercial and closed-source and can be queried through an API. It has a free tier where we can analyze documents using their pre-trained models for invoices and receipts, and gives us the bounding box and value of many relevant fields on the invoice. It includes an annotation pipeline where we can edit the output and correctly label the fields it has missed. The paid tier lets us train models and add new fields to the output, and we are able to retrain the model. FinanceDoc2JSON most likely uses similar methods like the ones used in Nanonets, but this is hard to say, as their solution is closed-source.

We have tested Nanonets manually with 18 different invoice layouts, and the model returns 15 different fields for every invoice, which means a maximum of 270 values. Not every invoice we tested with included every field, and the total fields that were included in the invoices was 206 fields. Out of these 206 fields, it did not identify or incorrectly identified 66 fields, which means it has an accuracy of 67.9%. It also had problems with the Figure 6.4.

Chapter 8

Discussion

8.1 Development method and process

In hindsight going for a full extreme programming approach might not have been the best choice. Usually the sprints had to be moved around some, and usually we would come up with new things we had to prioritize during the sprints. Therefore they would never be get fully settled.

However using an kanban board for dividing what to focus on was very helpful and made it very easy to know the state of the things.

We used the following boards.

- New issues
- Icebox (for issues put on hold)
- In progress
- Closed

All issues started as new issues, moved to icebox if it turned out not to be important. When a new sprint started we moved it to in progress, and then to closed when it was done.

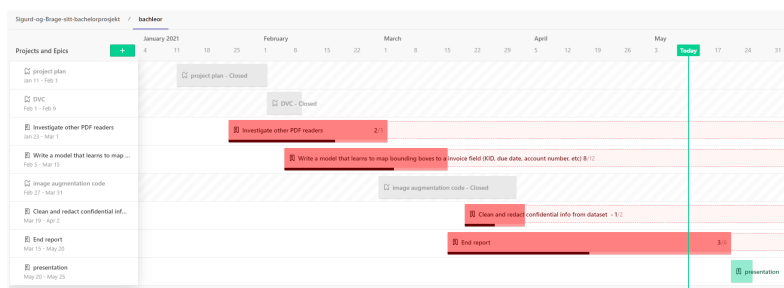


Figure 8.1: Our Gantt chart at the end of the project. The screenshot is from our project management framework called ZenHub [26], which is integrated with GitHub's issues.

8.2 Deviation from original project plan

We originally agreed to have weekly meetings. Because there not always was much to discuss after only a week, we decided instead to have meetings after bigger features has been implemented or to conduct Pull request (PR) reviews. This worked quite well.

As mentioned in section 8.1 we had also planned to do extreme programming and did that for some weeks, until we saw it was more beneficial to not have to big focus on the sprints. We still used planning pokers with our Kanban board, but still had a big focus on many on the coding standards side.

We also ended up spending not as many hours as planned on this project as seen in Appendix B. We believe we were working smart, and was still able to do a lot with the time we spent. However, because both of us are have full time jobs and are full time students there are not always that many hours left at the end of the day.

8.3 Other research

There has also been written multiple research papers on problems close to the one we are dealing with. For instance Attivissimo *et al.* [21] classifies the document as one of a set of known layouts, and extracts fields from identity documents. Their solution addresses the localisation of the document inside the image. We do not address this issue, because in most cases, the document is scanned and not being taken a picture of. Then OCR is performed on the document, and the text is extracted using a single-line text recognition model. This is possible because the layout of the document class they are dealing with do not change.

An approach we did not try is to use Natural language processing (NLP) to analyze the document. An approach like this would be to convert the documents to text, and then label our desired fields in the text using a Named Entity Recognizer (NER). We did not do this, as this required us to create a NLP dataset where we have labelled the invoice fields. We focused on the You only look once (YOLO) computer vision model instead.

LayoutLM by Xu *et al.* [27] uses both the words, the image of the word and the position of the word to classify each word. DocBank by Li *et al.* [22] is a large dataset of documents which can be used as training data in LayoutLM. DocBank was generated using Latex source files from arXiv to generate a large dataset of documents where every word is labeled into one of multiple categories such as abstract, authors, title, body and figures.

8.4 Improvements

There are a few things that are worth investigating further to make a even better model.

1. Add a word embedding layer like done in LayoutLM. This way the model could be able to better distinguish fields by looking at text close to the value.
2. Look at other backbones. Applying some attention mechanism would also be able to improve the performance.
3. Replace EasyOCR with a fully integrated solution. Running EasyOCR on top of our model has had some problems. Making the model fully integrated would make the model more stable as one could also optimize the actual output and not just the bounding box. Then one can train fully end to end.
4. Expand dataset with a synthetic dataset. Since some fields have a very uncommon value, there should be generated a synthetic dataset with a greater variation of currency and voucher types.
5. Complete the frontend or add option to write results to a database.
6. Combine the payment voucher and invoice model. Currently they are trained separately and use different configs, but combining them / using a shared backbone could improve the results for both models.
7. Both MOD10 and MOD11 can be used to validate kid and account numbers. There should be added an option to specify which to be used.

8.5 Comparison to other solutions

Creating an invoice understanding algorithm is hard and time consuming, and for a company which is restricted on time and resources such as Finexa, we would recommend to use services such as Nanonets[28], Google document ai [29] or Microsoft Form Recognizer [30] for higher accuracy. All of these have more experience with invoice understanding than us and have bigger infrastructure. Our model does better in some scenarios because our solution has support for the Norwegian's standard for KID and account numbers (which did not work with NanoNets and Google document ai when we tested them), but if these heavyweights, like Google or Microsoft wanted, they could easily remove the few advantages we have. There are some unique advantages with our solution by that it can be self hosted. Which could be an attractive reason for making sure no data is leaked. One could also much easier integrate fixes for the model, if it keeps classifying something wrongly. Having done this project gave us a greater understanding for how these solutions work.

8.5.1 Using Python as the language of the application

The Python language is very popular, and very flexible. However it is not strongly typed, we tried to combat this by using type checking software mypy, testing and linters to reduce the amount of bugs. However the strong typing software is not always the best either. On many lines we had to disable specific tests, because the linter did not pickup what we wanted to do. This added some noise to the codebase. We don't believe this affected us to much, but if the software is extended one should think about these things.

8.6 Stealing peoples job?

There was a big focus when working on this project to not steal other peoples jobs, but to supplement them. All outputs of the model should be verified by an human as discussed in section 5.7. The frontend that was started on for this projects build upon this idea. Creating an interface for making it easy to validate the model output is crucial both to improve the model, and to make sure the output are correct.

8.7 Authors relation to Finexa

Both authors are employed in Finexa's Corporate group, BA Group. Because of this, we have had to take extra care to discern when we represent Finexa and when we represent NTNU. We have not been paid or given any other type of reward to work on this project.

8.8 Toolchain / Software tools used

The we used GIT for tracking code changes, and integrated versioning and revision of the dataset and model using DVC¹. This way, we could track changes in our dataset alongside changes in our code base, and at the same time, keep large files out of the GIT repository.

The main software we used for machine learning was sklearn, numpy and pytorch.

To ensure code quality we used unit tests, pylint, mypy, and black code formatter.

We used Github as our issue tracker, with additional project management features, like epics, Gantt charts using Zenhub, which is free for academic purposes.

8.8.1 PyMuPDF

PyMuPDF[1] is a python package that has helped us at extracting PDFs information, such as words, text, bounding boxes, metadata and images. In theory, one could use this to "handcode" a program for extracting the relevant fields. However that probably not scale well and not be a general solution.

8.8.2 EasyOCR

By using EasyOCR, which is an open-source OCR engine, we can easily obtain the text in the image as well as their bounding boxes. EasyOCR works mostly with image files.

¹<https://dvc.org/doc/start>

8.8.3 Hardware used

We wanted to use an “Nvidia 3080 GTX” for this project, but because of the great GPU market shortage²³ we had to start with a basic GPU. We started with an “Nvidia GTX 780 3GB”. However, as we scaled up the model we needed a better GPU and upgraded to a better “Nvidia GTX 1080 Ti Armor OC 11 GB” at the end of March. The final hardware can be seen in Table 8.1.

CPU	AMD Ryzen 7 3700X
RAM	DDR4 16GB 2933MHz
GPU	GTX 1080 Ti Armor OC 11 GB

Table 8.1: Hardware used

²<https://www.theverge.com/2021/4/15/22385261/nvidia-gpu-shortage-rtx-3080-warning-comments-2021>

³<https://www.tek.no/nyheter/nyhet/i/weR1k4/ingen-bedring-i-sikte-for-nvidia-eller-amds-grafikkort>

Chapter 9

Conclusion

9.1 Result

Finexa wanted an OCR module that was able to parse different finance documents namely invoices, payment vouchers and receipts. Our solution is able to handle both invoices and payment vouchers, and correctly finds the fields on the invoice 57.96% of times. There is still room for improvement as discussed in section 8.4, but our solution is a good proof of concept.

Bibliography

- [1] J. McKie, *PyMuPDF: Python bindings for the PDF toolkit and renderer MuPDF*. [Online]. Available: <https://github.com/pymupdf/PyMuPDF> (visited on 16/05/2021).
- [2] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, 2016th ed. MIT Press. [Online]. Available: <https://www.deeplearningbook.org/> (visited on 15/05/2021).
- [3] M. A. Nielsen, 'Neural Networks and Deep Learning,' en, 2015, Publisher: Determination Press. [Online]. Available: <http://neuralnetworksanddeeplearning.com> (visited on 15/05/2021).
- [4] S. J. Russell and P. Norvig, *Artificial Intelligence: a modern approach*, 3rd ed. Pearson, 2009.
- [5] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors,' en, *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, Number: 6088 Publisher: Nature Publishing Group, ISSN: 1476-4687. DOI: 10.1038/323533a0. [Online]. Available: <https://www.nature.com/articles/323533a0> (visited on 13/05/2021).
- [6] *A Neural Network in 11 lines of Python (Part 1) - i am trask*. [Online]. Available: <http://iamtrask.github.io/2015/07/12/basic-python-network/> (visited on 15/05/2021).
- [7] *CS231n Convolutional Neural Networks for Visual Recognition*. [Online]. Available: <https://cs231n.github.io/convolutional-networks/> (visited on 19/05/2021).
- [8] *4. Major Architectures of Deep Networks - Deep Learning [Book]*, en. [Online]. Available: <https://www.oreilly.com/library/view/deep-learning/9781491924570/ch04.html> (visited on 15/05/2021).
- [9] M. S. Mudaragadda, *Max Pooling in Convolutional Neural Network and Its Features*, en-US, Mar. 2020. [Online]. Available: <https://analyticsindiamag.com/max-pooling-in-convolutional-neural-network-and-its-features/> (visited on 18/05/2021).
- [10] A. Karpathy, *Software 2.0*, en, Mar. 2021. [Online]. Available: <https://karpathy.medium.com/software-2-0-a64152b37c35> (visited on 12/05/2021).

- [11] *Tesla hires deep learning expert Andrej Karpathy to lead Autopilot vision* | *TechCrunch*. [Online]. Available: <https://techcrunch.com/2017/06/20/tesla-hires-deep-learning-expert-andrej-karpathy-to-lead-autopilot-vision/> (visited on 15/05/2021).
- [12] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'ImageNet classification with deep convolutional neural networks,' en, *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386. [Online]. Available: <https://dl.acm.org/doi/10.1145/3065386> (visited on 15/05/2021).
- [13] Databricks, *Building the Software 2.0 Stack (Andrej Karpathy)*, Aug. 2018. [Online]. Available: <https://www.youtube.com/watch?v=y57wwucbXR8> (visited on 18/05/2021).
- [14] Tesla, *Tesla Autonomy Day*, Apr. 2019. [Online]. Available: <https://www.youtube.com/watch?v=Ucp0TTmvq0E&t=7714s> (visited on 18/05/2021).
- [15] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, 'Gradient-based learning applied to document recognition,' *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, Conference Name: Proceedings of the IEEE, ISSN: 1558-2256. DOI: 10.1109/5.726791.
- [16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting,' *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: <http://jmlr.org/papers/v15/srivastava14a.html>.
- [17] S. Ioffe and C. Szegedy, 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,' *arXiv:1502.03167 [cs]*, Mar. 2015, arXiv: 1502.03167. [Online]. Available: <http://arxiv.org/abs/1502.03167> (visited on 13/05/2021).
- [18] K. He, X. Zhang, S. Ren and J. Sun, 'Deep Residual Learning for Image Recognition,' *arXiv:1512.03385 [cs]*, Dec. 2015, arXiv: 1512.03385. [Online]. Available: <http://arxiv.org/abs/1512.03385> (visited on 13/05/2021).
- [19] K. Simonyan and A. Zisserman, 'Very Deep Convolutional Networks for Large-Scale Image Recognition,' *arXiv:1409.1556 [cs]*, Apr. 2015, arXiv: 1409.1556. [Online]. Available: <http://arxiv.org/abs/1409.1556> (visited on 13/05/2021).
- [20] R. Munroe, *Machine Learning*. [Online]. Available: <https://xkcd.com/1838/> (visited on 18/05/2021).
- [21] F. Attivissimo, N. Giaquinto, M. Scarpetta and M. Spadavecchia, 'An Automatic Reader of Identity Documents,' en, in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Bari, Italy: IEEE, Oct. 2019, pp. 3525–3530, ISBN: 978-1-72814-569-3. DOI: 10.1109/SMC.2019.8914438.

- [Online]. Available: <https://ieeexplore.ieee.org/document/8914438/> (visited on 12/05/2021).
- [22] M. Li, Y. Xu, L. Cui, S. Huang, F. Wei, Z. Li and M. Zhou, 'DocBank: A Benchmark Dataset for Document Layout Analysis,' *arXiv:2006.01038 [cs]*, Nov. 2020, arXiv: 2006.01038. [Online]. Available: <http://arxiv.org/abs/2006.01038> (visited on 13/05/2021).
- [23] O. Ronneberger, P. Fischer and T. Brox, 'U-Net: Convolutional Networks for Biomedical Image Segmentation,' *arXiv:1505.04597 [cs]*, May 2015, arXiv: 1505.04597. [Online]. Available: <http://arxiv.org/abs/1505.04597> (visited on 09/02/2021).
- [24] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros, 'Image-to-Image Translation with Conditional Adversarial Networks,' *arXiv:1611.07004 [cs]*, Nov. 2018. [Online]. Available: <http://arxiv.org/abs/1611.07004> (visited on 16/05/2021).
- [25] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, 'You Only Look Once: Unified, Real-Time Object Detection,' *arXiv:1506.02640 [cs]*, May 2016, arXiv: 1506.02640. [Online]. Available: <http://arxiv.org/abs/1506.02640> (visited on 13/02/2021).
- [26] ZenHub, *ZenHub - Agile Project Management for GitHub*, en, 2021. [Online]. Available: <https://www.zenhub.com/> (visited on 10/05/2021).
- [27] Y. Xu, Y. Xu, T. Lv, L. Cui, F. Wei, G. Wang, Y. Lu, D. Florencio, C. Zhang, W. Che, M. Zhang and L. Zhou, 'LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding,' *arXiv:2012.14740 [cs]*, Dec. 2020, arXiv: 2012.14740. [Online]. Available: <http://arxiv.org/abs/2012.14740> (visited on 02/05/2021).
- [28] *Intelligent document processing with AI*, en. [Online]. Available: <https://nanonets.com> (visited on 19/05/2021).
- [29] *Document AI Solution*, en. [Online]. Available: <https://cloud.google.com/document-ai> (visited on 19/05/2021).
- [30] Microsoft, *Invoices - Form Recognizer - Azure Cognitive Services*, en-us. [Online]. Available: <https://docs.microsoft.com/en-us/azure/cognitive-services/form-recognizer/concept-invoices> (visited on 19/05/2021).

Glossary

Accounts Receivable Management (*Fakturaoppfølging*) Also called Invoice follow-up. Ensures customers pay their money on time and reduces the risk of any bad debts, by reminding customers and collecting the money on time. Additionally identifying the reasons for delays and finding solutions to related issues.¹. 2

Invoice (*Faktura*) An official document relating to a transaction between a buyer and a seller, which documents how much is owed.². 1

long tail From statistics. The long tail is the part of the distribution having many occurrences that are far away from the mean³. 15

Payment Voucher (*Bankbilag*) A document generated by the bank, which tells how much money was transferred for an invoice.. 1, 61

¹<https://cleartax.in/s/accounts-receivable-management>

²<https://en.wikipedia.org/wiki/Invoice>

³<https://venturebeat.com/2020/08/14/how-to-improve-ai-economics-by-taming-the-long-tail-of-data/>

Acronyms

JSON JavaScript Object Notation. 17

KID Kundeidentifikasjonsnummer. 1, 62

NER Named Entity Recognizer. 64

NLP Natural language processing. 64

OCR Optical Character Recognition. 1, 3, 64, 66

PDF Portable document format. 1–3, 66

PR Pull request. 64

wandb Weights & Biases. x, 31, 51, 60

YOLO You only look once. 64

Appendix A

Meeting logs

Some short summary of the meetings taken place each month with some of the big milestones.

11.01.2021

Kick-start meeting with Finexa. Got access to dataset and some discussion about the problem in general.

A.0.1 15.01.2021

First meeting with Tom. We had some questions regarding the project plan, but there was also some general talk about the project.

Tom advised us to research NTNU open for similar projects. He also had some feedback regarding the group rules and advised us to enforce some stricter rules.

23.01.2021

First real meeting between Sigurd and Brage regarding the code and architecture choices.

Agreed on

- One model for finding the fields.
- One model for reading out the text.

03.02.2021

Second meeting with Tom, got feedback regarding project plan. Before the meeting we sent over our project plan draft and the agenda for the meeting was to get some feedback. We got some good feedback to further improve the project plan.

07.02.2021

Discussed the following pull request and did some pair programming.

22.02.2021

Agreed to instead of having meetings to send each other short updates on discord at the end of each sprint.

A.0.2 08.03.2021

Short status meeting. Agreed on some frontend design and the idea of implementing batch validation by for instance making sure there is no jump in the invoice numbers.

A.0.3 06.04.2021

Longer status meeting. Discussion of the status of the frontend and the backend.

29.04.2021 - 02.05.2021

Daily meetings to complete a draft of the report. Creating a agenda to be able to get through the last month. We agreed to put all the time into the report the next week, and use the remaining time to do some last experiments (looking especially at ways for making the tesseract output more stable).

A.0.4 06.05.2021

Meeting with Tom to discuss report draft. We could constructive feedback that we will implement into the report to make it very nice.

Appendix B

Timesheet

B.1 Short summary for each month

Info	January	February	March	April	May
What we did	We had kick off meeting with Finexa and Tom. Wrote the group guidelines and started on the project plan. Did a small amount of coding by writing the first validation class for KID.	Updated the project plan based on feedback. Started to investigate other models, and other research.	Worked a lot on the dataset, analyzing it and started to train a Yolo model on it.	Added support for bankvouchers, and scripts for fetching more data with high loss. Started to work on report.	Report, report, and report.

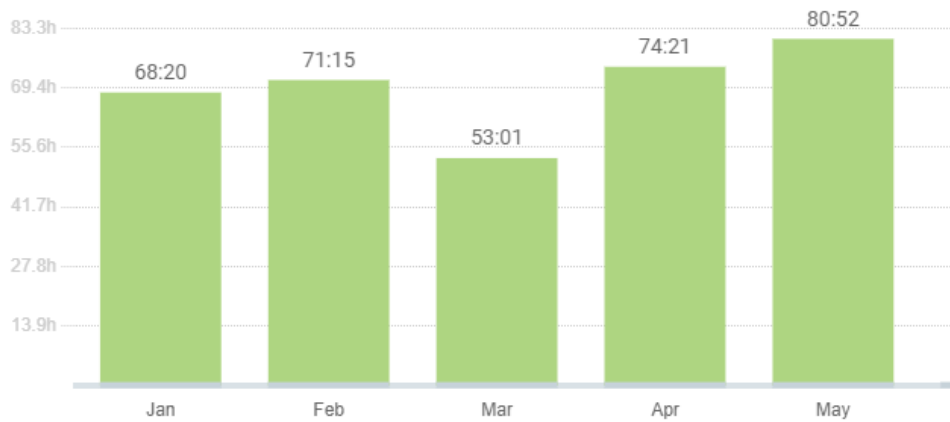
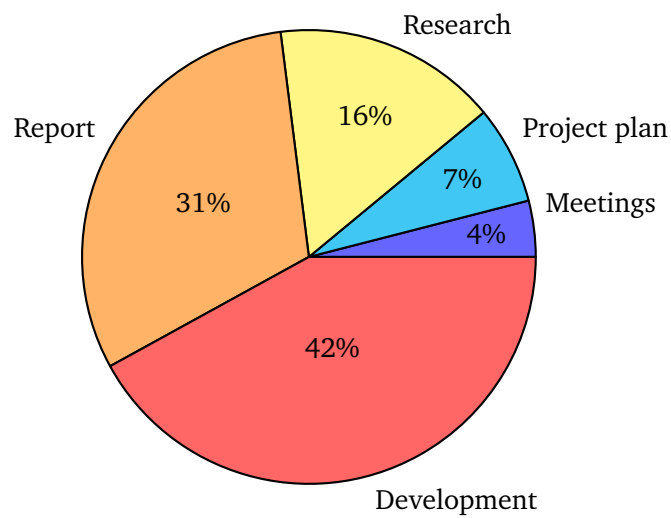


Figure B.1: A summary of hours worked per month combined for all group members

B.2 Division of time between tasks



B.3 Github activity

As one can see we had weekly commits to our repositories.



(a) Commits over time for our main repository. Some commits are missing because of the way Github assigns a commit to an author based on email.



(b) Commits over time for our model-zoo / research repository

Figure B.2: The GIT commit activity

Appendix C

Assignment description

Automatisert bilagsgjenkjenning

Om Finexa

Finexa Norge AS er leverer tjenester innen fakturaoppfølging og inkasso, men leverer også en god del egenutviklede løsninger som skal gjøre regnskaps-relaterte oppgaver lettere for våre oppdragsgivere.

Finexa har ikke gjennomført bacheloroppgaver tidligere, men totalt har konsernet ansatt 3 studenter fra NTNU i Gjøvik. Finexa har hovedkontor i Drammensveien 20, 3300 Hokksund, men IT avdelingen holder til i Torggata 72, 2317 Hamar.

Problemstilling

Finexa har integrasjoner med de fleste regnskapssystemer som er i bruk i Norge i dag. I forbindelse med arbeidet i Finexa oppstår ofte behovet for å kunne gjenkjenne data i PDF fakturaer og kunne levere disse på en strukturert måte (JSON). I noen tilfeller fungerer ikke integrasjonen mellom regnskapssystemet godt nok av ulike årsaker. Enkelte regnskapssystemer har innebygd funksjonalitet for automatisk gjenkjenning av fakturaer mens en god del av de eldre systemene ikke har det. Ved å utvikle denne typen funksjonalitet kan vi tilby våre kunder en automatisert bilagsflyt selv om regnskapssystemet ikke støtter dette som standardfunksjonalitet.

Vi ønsker en fleksibel OCR modul som kan lese følgende og automatisk skille mellom følgende typer bilag:

- PDF Faktura
 - En PDF fil som er produsert av et regnskapsprogram. Denne er vanligvis direkte produsert, men i enkelte tilfeller vil den også kunne være printet og scannet igjen slik at det vil være en del støy i dataene.
- Betalingsbilag fra banken
 - Dette er informasjon om innbetalinger som kommer fra bankene. Vi har integrasjon med mange forskjellige banker som har forskjellig layout på disse bilagene. Disse vil alltid komme som produsert PDF filer (ingen støy).
- Kvittering
 - Dette kan være produserte PDF bilag men er som oftest scannede bilder av kvitteringer. Her vil det vanligvis være en god del støy med i dataene.

Det er ønskelig å kunne lese ut mest mulig komplett informasjon fra disse dokumentene slik som:

- Beløp
- Fakturadato
- Forfallsdato
- Kundenummer
- Fakturanummer
- KID

- Kreditor (den som fakturaer og skal motta betaling)
 - Navn
 - Adresse
 - Orgnr. hvis det er et foretak.
- Debitor (den som skal betale fakturaen)
 - Navn
 - Adresse
 - Orgnr. hvis det er et foretak.
- Kontonummer

Det er ønskelig at løsningen struktureres slik at det er mulig å kalle den via et HTTP api hvor bilaget sendes inn via en HTTP POST metode (enten som PDF eller bildefil) og den strukturerte informasjonen kommer i retur som JSON. Studentene står helt fritt til å velge den fremgangsmåten, programmeringsspråkene og utviklingsverktøyene som de selv synes er hensiktsmessige. Studentene står fritt til å legge vekt på de områdene innenfor oppgaven som de anser som relevant.

Utfordringen med denne oppgaven er naturligvis at fakturaer ofte har veldig forskjellig utseende og oppsett. Det som er interessant å vite er hvor høy treffprosent som er mulig å oppnå og hvor pålitelig og tolerant mot støy og endringer i layout, en slik OCR-løsning vil være.

Datasett

Finexa Norge AS har i dag følgende datagrunnlag som kan brukes for å teste og optimere gjenkjenningen:

- 1,2 millioner PDF fakturaer
- Over 50 000 bankbilag

Da dette er reelle fakturaer er Finexa Norge AS nødt til å ta forbehold om at konkret kundeinformasjon som er inneholdt i fakturaene behandles konfidensielt av studentene. Resultater og tekniske detaljer trenger derimot ikke å behandles konfidensielt.

Vi tenker at oppgaven passer best for studiet BIDAT Bachelor i Ingeniørfag- data.

Finexa Norge AS | Org. Nr.: 921 229 216

Kontaktperson:

Felix Schoeler

E-post: felix@finexa.no

Tlf: +47 984 61 290

Appendix D

Project plan

Project plan for Bachelor's project

Bank related document reading

Brage Arnkværn Sigurd Schoeler
NTNU Gjøvik NTNU Gjøvik
bragearn@stud.ntnu.no sigurssc@stud.ntnu.no

January 2021

Contents

1	Introduction	2
1.1	The task	2
1.2	Goals	2
2	Scope	3
2.1	Functionality	3
2.2	Code	3
2.3	Machine learning model	3
3	Implementation	4
3.1	Milestones	4
3.2	Evaluation	5
4	Planning and reporting	5
4.1	Development model	5
4.2	Status meetings	7
5	Project organization	7
5.1	Responsibilities	7
5.1.1	Brage	7
5.1.2	Sigurd	7
6	Risk assessment	8
6.1	Critical areas that defines success	8
7	Quality assurance	9

8 Resources	9
8.1 Time	9
8.2 Schedule	9
8.3 Cost and budget	10
A GPU budget	11
B Input and output of segmentation model	12
C Sequence and UML diagram	13

1 Introduction

This document is an outline about our bachelor project. It describes ideas for how we are going to approach the task and how we envision the process will be. This document includes the goal, scope of the project, how we plan to implement things and how we have considered things like positional costs.

1.1 The task

The research question in our project will be finding out how to identify and extract key facts from scanned finance/bank related documents. The task was given to us by Finexa AS¹ and they were interested in extracting data from invoices, receipts and other vouchers. Our solution should be able to accurately return the data in a machine readable format, ideally in a key/value format (like JSON), to make it easy to query the data.

1.2 Goals

Our main goal is to be able to create a solution that is able to extract key facts from both documents that are digitally created and transmitted, scanned and photographed documents. In other words the source of the input should not matter.

We also want to take on a challenge and make the solution universal. Therefore it should be able to work with multiple different types of documents like invoices, receipts and vouchers. It should also be general enough that it will work with all the different types of formats that exists for these documents.

It's also important for us that the information that is returned from our system is accurate. We have multiple ideas for making sure this goal is reached See section 7 and section 6.

¹Organization number 921229216

2 Scope

Because OCR in general is looked at as a solved problem, our main job is to come up with a way to map the different keys to the values based on a image input.

Therefore we will probably leverage a “pretrained” OCR model with our own model in the beginning. The main job of our model is to be able to detect where the different keys and values are, then the OCR model extracts the text. If we have time, we will also use our own custom OCR model for optimizing everything. This is further discussed in 2.3.

Because we will probably have some problems with reading the invoice lines and having different properties match up correctly. We will probably have to move to our own OCR model after some of the first challenges are solved. By having our own fully end to end model we will probably get higher accuracy.

2.1 Functionality

2.2 Code

Most of the code will be logic for validating the different invoice fields, retrieving the PDF/JPEG and code for interacting with the model and getting out the results. There should also be created some basic GUI.

For instance there has to be logic for handling bad input to the model. One example of a bad input is a blurry image. There should be logic for sharpening all images. It also have to handles cases when the model is not sure what to do (low confidence). Then our code has to report back to the user that the results might be incorrect.

We also need scripts for creating the dataset used to train the model that has to do image augmentation and creating bounding boxes for training the model.

In appendix C we have attached sequence and UML diagrams for how we expect the project to evolve. It includes the main classes and their relation.

2.3 Machine learning model

By looking at the problem as an object detection / segmentation problem, we should be able to leverage a the same model architecture used for these problems in our model.

The input to the model is a rasterized image of an invoice or receipt. Then the model should find the location of the field inside the image.

The idea is that the model should output a mask so we can extract only the relevant field and give that to an OCR model. Something like in 2, the model is queried to extract the KID and will remove everything that is not related to the kid.

Most segmentation models are nowadays some form of autoencoder, instead of reinventing the wheel we would probably build upon the U-Net[1] architecture

which is a segmentation model that has gotten high accuracy on segmentation tasks. YOLO[2] is also an architecture which is popular for object detection. Both of these architecture are worth investigating to see if could be applied to our problem.

3 Implementation

3.1 Milestones

To break down the creation of our solution, we have split the project into multiple Epics, where every epic marks the finish of a feature. The epics are further split up into weekly sprints with a milestone at the end of every sprint.

Because there is a lot of uncertainty in how our solution will be implemented, it is hard to break down the epics into concrete issues before we have researched how we are going to implement the solution. We have identified a cycle for creating Epics, which approximately break down into the following:

1. Research how similar problems have been solved as inspiration and write project plan based on this.
2. Implement our solution based on the research in the previous epic
3. Evaluate our current method.
4. Research other solutions.

The idea is to break each problem into that cycle.

For every time we go through this cycle, we will have to research and fine-tune our solution, and we will end up with multiple possible solutions, of which the best one will be used. We will also keep logs of all our experiments, the code and dataset state. W&B for logging experiments and DVC + GIT for dataset and code state. We have also identified a few of the epics we have to overcome to get our solution, but the project has one big moving part, the machine learning model. In addition to the existing epics, we will gradually improve this model and create many sub issues for the model. For instance one of the first epic is for the model to learn where the KID is.

1. Create the first version of the dataset and version control it with DVC.
2. Transform the dataset with the different bounding boxes and map these to values. This is crucial for getting started with the implementation of the machine learning model and is something we have to do early.
3. Write code for doing KID validation, this could be used to aim the model.
4. Write a model that learns to map bounding boxes to a invoice field (KID, due date, account number, etc)

5. Being able to read the first kid. There are many fields our model has to be able to find, however our first milestone is to be able to read the KID. The reason we choose KID is because it does not have a standard location and it can easily be verified by the check digits.
6. Being able to read the first invoice number. Same reason as the KID, it has check digits which makes it a bit easier to find.
7. Write code for doing image augmentation to prevent over-fitting and to generalize the solution.
8. Research other new solutions and implement them to progressively be able to read new fields

3.2 Evaluation

Given that OCR is mostly a solved problem. We can leverage commoditized technology to kick-start the project, and helps us so we don't have to reinvent the wheel. A lot of invoice data has error detection codes, and we can leverage these to know if the model has failed to read the data. If our solution detects these bad cases, it should not give out wrong info, or at least notify the user that there is something wrong or give alternative suggestions.

4 Planning and reporting

4.1 Development model

We will use a development model that builds upon ideas from “Agile” and “Extreme Programming”. We will also use Zenhub as the project management tool which is a tool for making working with Agile easier and it integrates directly with GitHub.

Principle / Concept	Description	How we will use it
The backlog	List of work to be done	This contains the requirements for the final release, what we work on each week will be based on the backlog and will change with time.
User stories	Stories describes what the end user want. Finexa has given us an description of how they expect the solution to work. These will be added as stories.	We will implement stories as issues, since we won't have so much talk to the end user the stories will mostly describes things we have to implement and follow the SMART acronym.
Sprints	Taking stories out of backlog and working on them	We will have one week sprints, they might be extended if the tasks from the previous week took longer to implement. We will take the most important items out of the backlog and work on them.
Product manager role	In Agile and XP the product manager is the one who controls the backlog and sprints.	Sigurd is the project manager, and has the responsibility to have a high-level overview of the progress of the project
Daily stands up	In Agile it's normal to have daily discussion about the status of the sprint.	We will give each other a status update on discord each time something big happens. We will also have at least one or two weekly meetings.
Test driven development	Create tests first. Tests should run fast (less than 1 min). Refactor after a feature has been finished.	We will implement tests and run these on every push.
Continuous integration	Continuous help with maintain a healthy code-base and is especially popular in XP	We will have a CI that runs linters, unit tests and code formatters to ensure that all code changes can be released.

Code refactoring	Afer the code is written it has to be refactored to be clean code.	We will use mypy to enforce types to make refactorign easy and unit tests to make sure no refactoring break the exsisting code.
Releases	Small and frequent releases in contrast to big updates	We will also aim for many small releases, the idea is that the code in master should always work and could be sent over to Finexa at anytime if they wanted to test the current state of the project.

4.2 Status meetings

We will have weekly status meetings where the main goal is to reflect on what has been done the previous week, and plan for the next week and beyond. We will also go through maintenance tasks that has to be attended to continually, like maintaining project structure, code quality assessment and reviewing issues and pull requests. Given that we are in a pandemic most status meetings will be held digitally.

5 Project organization

5.1 Responsibilities

Role	Name
Client	Finexa AS
AI Scientist	Brage
Project Manager	Sigurd
Supervisor	Tom Røise

All the members have the responsibility to contribute towards the progress of the project, and should feel responsible for the product they create. Other obligations are outlined in

5.1.1 Brage

Researching and training the machine learning model.

5.1.2 Sigurd

Project management, status, reporting, logging, organizing meetings.

6 Risk assessment

Risk	probability	Consequence	Description	Countermeasures
Too little time	Medium	High	We might underestimate the time we have to finish the project, and might run out of time before the solution is properly finished and refined.	Proper project management and plan epics and milestones ahead of time
Uncertain Complexity	Medium	Medium	There is a big cone in uncertainty in how technically difficult the final solution will be, and we do not know exactly how the solution will look.	We will start the project with an evaluation of other similar solutions and will look at if these solutions could be used directly in our solution, or if we could take ideas from these other solution.
Requirement changes	Small	Medium	The requirements of the solution have been given in written form and the requirements give us a large amount of freedom as for how the solution should look, as long as the output is correct.	None
Poor Quality Code	High	Medium	We acknowledge that there is a risk of ending up with code that has poor quality, and is cluttered with lots of bugs.	We will use CI pipeline (linters, test and formatters) to ensure that code is being developed with a high quality the first time, and that it is being tested and reviewed properly.

6.1 Critical areas that defines success

To be able to have a successful project we need to be able to be able to create a model that can link keys and values. Then we also need to be able to extract the values correctly.

Given that OCR is mostly a solved problem. We can leverage commoditized technology to kick-start the project, and helps us so we don't have to reinvent the wheel. A lot of invoice data has error detection codes, and we can leverage these to know if the model has failed to read the data. If our solution detects these bad cases, it should not give out wrong info, or at least notify the user that there is something wrong or give alternative suggestions.

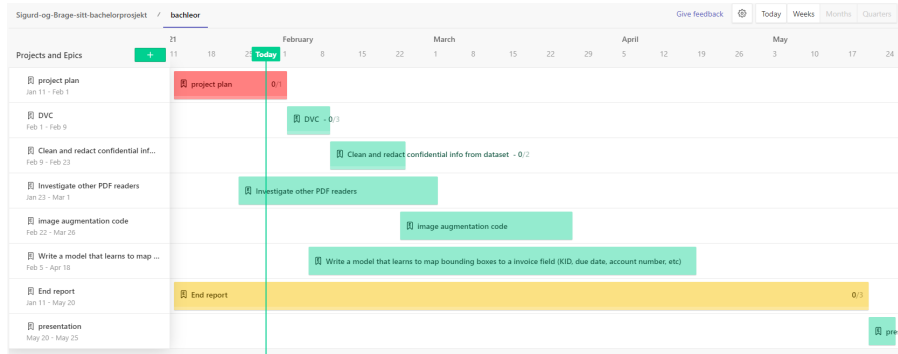


Figure 1: The Gantt diagram which is based on the issues at the beginning of the project

7 Quality assurance

Given the nature of this problem it's easy to quantify how well our solution is performing (by utilizing a test dataset and measure the accuracy).

We also want the system to work as well with scanned documents as the once generated by a computer. By creating a dataset which includes both types, we should be able to create a solution with high quality.

8 Resources

To keep the project under control we have added some constraints to make sure we have equal effort and that no unpredictable cost appears.

8.1 Time

Both are expected to work around 25 hours each week. This will be enforced by using Clockify, issues and GIT activity for tracking all changes. This way it's easy to see if someone is behind on the time-sheet. This is also enforced in the group guidelines and will be reviewed each week.

We will also use milestones in the issue tracker. This way we can plan for weeks were we have to put in extra effort if we are behind a milestone. We assume this will happen as the project evolves.

8.2 Schedule

The gantt chart is based on the issues we have created in our github repository and is used to track our ongoing efforts and will be updated as we work on the project.

8.3 Cost and budget

There is some other services that should be considered, for instance:

Service	Use-case	Price	Total (max) price
GPU usage	To be able to train our models quickly.	250\$, see Appendix A for details	250\$
wandb	Logging of experiments	0\$ for a shared account, 35\$ for teams account	140\$
Github team pro	Gives us some more features for making it easier control the git repository.	8\$ each month	36\$
PDF manipulation software	By using a software like Adobe acrobat pro we will be able to remove metadata and edit the content of the PDF. This is useful to anonymous real invoices.	20\$ each month. libraries like pymupdf ^a and by doing this not having any costs. We have not finalized this decision yet. ^a https://pymupdf.readthedocs.io/en/latest/	80\$
Cloud storage	Storage of dataset and models. We also need storage space for DVC which will track both of theses.	Both Azure[3] and S3[4] has a gigabyte price of around 0.23\$. We would probably never need more than 100 GB (the current dataset has a size of 2.3 GB). Resulting in 23\$ each month.	92\$
Sum			598\$

A GPU budget

To test our machine learning models we can use free services like Kaggle / Google Colab (which have more than 30 hours of free GPU time each week) and see if an idea or experiment is going well by looking at the loss function. Because of limitations in these free services, we will have to use paid services when we have to train the model for longer time intervals and/or on a larger dataset.

There are a few free services we could use for kick starting the project. Both Kaggle and Colab offers free GPU usage, but to scale our GPU usage, we could use a renting service like Vast.ai or Lambda GPU Cloud, were we pay on-demand pricing for renting a GPU. Here we pay around 1.4\$ for each teraflop,

The GPU will most likely be what haves the highest cost of the project. We believe we are likely to spend around 300\$ on this.

B Input and output of segmentation model

Below is an example of a model input and model output. Input is an invoice and the model is asked to extract the kid. The model outputs only the kid field. Then a OCR model will pickup the output to retrieve the field value.

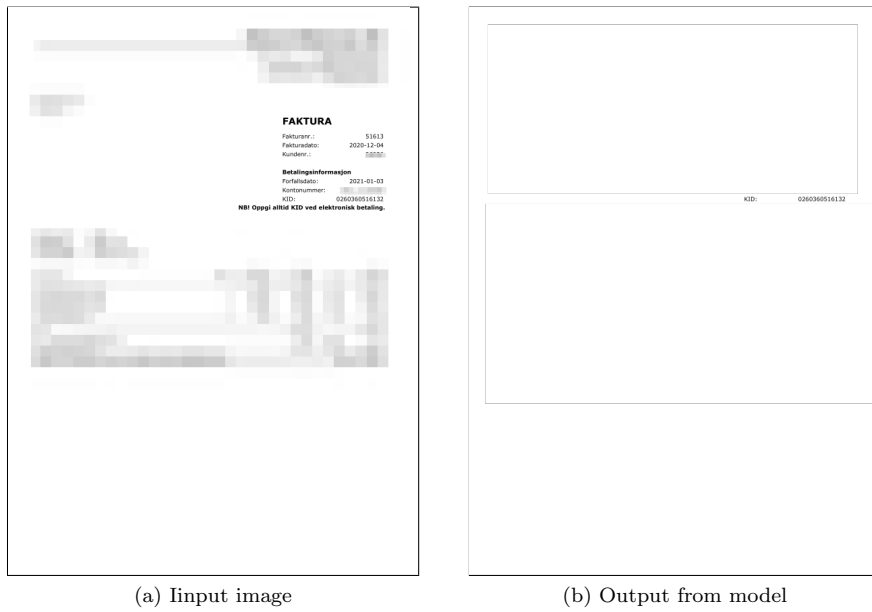
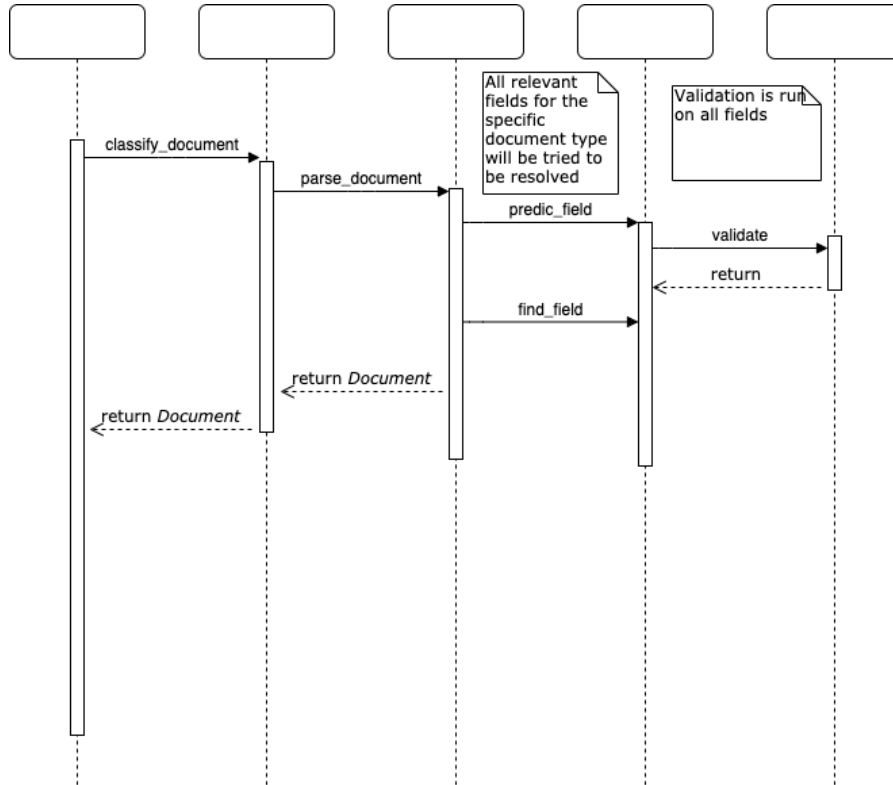


Figure 2: Input and output from model

C Sequence and UML diagram



As can be seen in Figure 3 we will use abstract classes a lot. This makes it easier to test the code and makes sure objects that look alike (receipts and invoices) has a common interface.

This can also be seen in the sequence diagram. Since the application will try to resolve each field both by looking at the text content directly off the pdf and what the models "sees" from the image. The same is true for validating the fields, all fields will have a validate method and it will be called when a document is retrieved.

More methods will probably be added, but we think the general structure will look like the UML diagram.

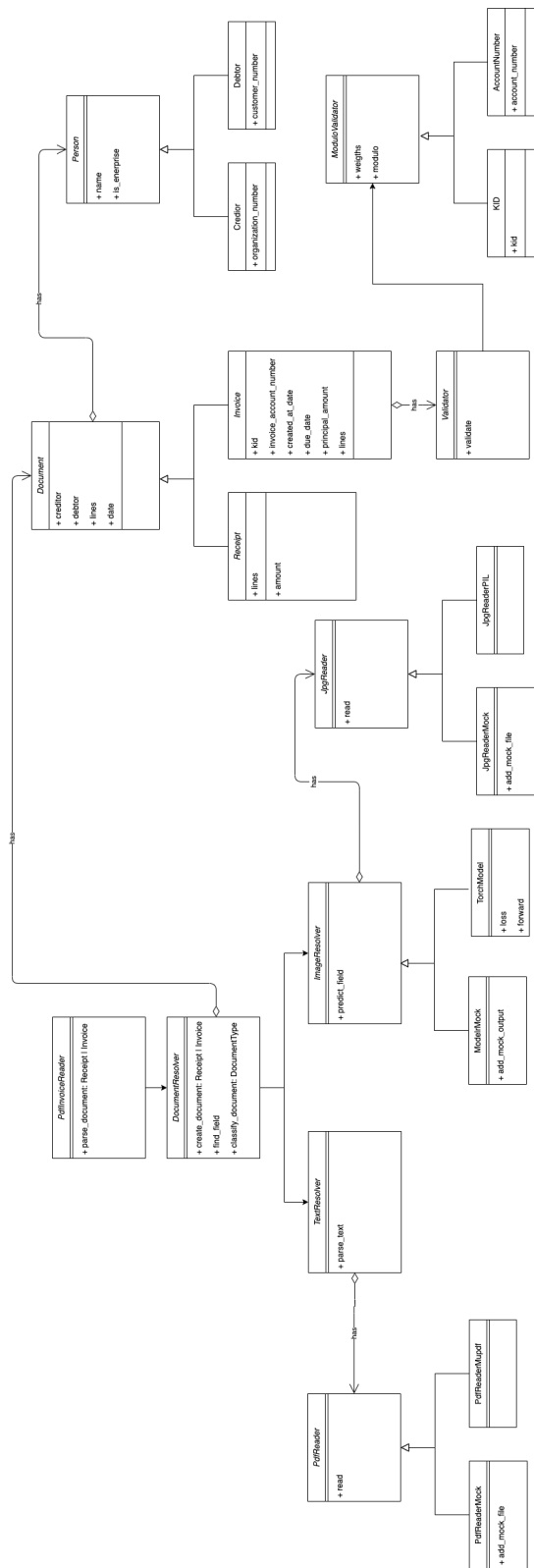


Figure 3: Uml diagram

References

- [1] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. arXiv: 1505.04597 [cs.CV].
- [2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, 2016. arXiv: 1506.02640 [cs.CV].
- [3] (). “Azure storage blobs pricing — microsoft azure,” [Online]. Available: <https://azure.microsoft.com/en-us/pricing/details/storage/blobs/> (visited on 01/23/2021).
- [4] (). “Amazon s3 simple storage service pricing - amazon web services,” Amazon Web Services, Inc. [Online]. Available: <https://aws.amazon.com/s3/pricing/> (visited on 01/23/2021).

Appendix E

Project agreement (Prosjektavtale)

Prosjektavtale

mellom **NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik** (utdanningsinstitusjon), og

Finexa Norge AS (oppdragsgiver), og

Sigurd Santiago Schoeler og **Brage Arnkværn** (studenter).

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 11.01.2021 til 20.05.2021 .

Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
 - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon, reiser og nødvendig overnatting på steder langt fra NTNU i Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
 - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.
3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.
4. Alle beståtte bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv NTNU Open.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.
6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.
7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.
8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.
9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.
10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.
11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): Tom Røise

Oppdragsgivers kontaktperson (navn): Felix Schoeler

Student(er) (signatur): SIGURD S. dato 13.01.2020

Bruse t dato 13.01.2020

Oppdragsgiver (signatur): Felix Schoeler dato 13.01.2020

*Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.*

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.

Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato _____

Appendix F

Group guidelines

1 Bachelor: Group guidelines

1.1 Strive for clean code

Read "Clean code" by Robert C. Martin, and use the principles described in this book to create clean code. Don't be doctrinaire.

1.2 Minimize knowledge gap

If someone knows more than the other, or has specialized in a field, knowledge should be actively exchanged. This is faster than researching yourself, and gets all the members on the same page.

1.3 Document getting started

Write documentation for how to clone, install, set up, configure the project, run the tests, run the top-level code and deploy the code base.

1.4 Be ahead of schedule

Use GitHub issue tracker actively to track how well deadlines are met.

1.5 Equal effort

All members should end with roughly the same amount of effort, time and contribution towards the project.

1.6 Use English

Write code and reports in English.

Sigurd Schoeler:

SIGURD S.

Brage Arnkværn:

Brage t

Date: 13. January 2021

