Final Product DASC2

Lukas Greven, Felix Pralle, Niklas Mezynski

January 2024

Contents

1 Introduction **1**
1.1 Description of the business case 1
1.2 Project and research Goal 1

2 Project realization **1**
2.1 Dataset 2
2.2 Pre-processing 2
2.3 Training 2
2.4 Grid Search 4
2.4.1 Grid Search 1 4
2.5 Grid Search 2 4
2.6 Grid Search 3 5

3 Results **5**
3.1 YOLOv8x Model 5
3.1.1 YOLOv8m Model 6
3.1.2 YOLOv8m Model 32 Epochs 6

4 Conclusion and recommendation **7**

1 Introduction

1.1 Description of the business case

Our Business case is about digitizing menus for restaurants. The owners of said restaurants do not want to waste their time trying to figure out how to do that. The solution would be to have an app which would be able to convert pictures of a menu to a computer readable file. The then created file could be used to create an online menu which would allow other users to easily see the variety of dishes the restaurant offers.
Our first Idea was to use an OCR which would give us the text of the picture and then use a NLP model to create a JSON file of it. NLP models are very expensive in terms of time and computation power so we asked ourselves if there would be a simpler solution to this problem. Also, this figured out to be too simple for this assignment as there would not have been a part where we would train a model ourselves. The solution for that would be to train a model to recognize certain parts of the menu to then categorize the text on the menu and therefore break down the issue. The then classified parts of the menu could then be used for the OCR so that it can give us the text of said part which would enable us to structure the computer readable file much better.

1.2 Project and research Goal

The goal of our trained model is to recognize different parts of a menu. Our focus was that the model is able to differentiate between *sections*, *category titles* and *menu items*. Sections consist of one category title and multiple menu items, e.g the category title of the section is pizza and the items would be a margheritha for example with its price. The research goal is to find out if it is possible to train a object detection model that detects certain areas in a graphical layout. The main issue could be that the model is able to identify the text but cannot categorize it properly as for example categories and items are just text which could lead to confusion for the model. But therefore we have the benefit of having the three classifiers, so even if the model is not able to detect all classifiers on the menu at once. We can train it firstly to detect sections as those should be easily distinguishable. When that is done we know that there is a high probability that there is a category within that section which should make it easier for the model to find it. Also the items should then be easier to identify for the model as it does not have to differentiate between sections or items anymore.

2 Project realization

This section describes the steps that were taken throughout the research project.

Class	Average No. of Bounding Boxes
section	3.344
category_title	3.069
menu_item	18.425
sum	24.838

Table 1: Average Number of Bounding Boxes per image

2.1 Dataset

Before training any model, a dataset for the described use case was required. As there was no publicly available dataset, labeling a custom dataset was required. This was achieved by building a web scraper that would look up the term "menu card" on google images and download the first 200 images into a local folder. After removing low quality images and duplicates, 90 images were left for training. A tool called "V7 Darwin" (`https://darwin.v7labs.com/`) was used to draw the bounding boxes of the three different classes on the images. When all images were labeled, they were exported to the YOLO format, which outputs a txt file for each image with all of its bounding in the following format $(class\_id, x_{center}, y_{center}, width, height)$. Table 1 shows the average of how many of each class there is per image.

2.2 Pre-processing

In order to train the model with the training data, all the images had to be scaled to the same size. Therefore some utility functions were written that would rescale the images and its corresponding bounding box txt files. Also all images without bounding boxes have been filtered out. For this particular use case it was decided scale the images by stretching (or squeezing) them into the target size of $512px \times 512px$. By stretching the image, the actual style of the layout does not get modified. Also this technique could help blurring the fonts itself which works against the risk of the model recognizing text only instead of the actual layout.

2.3 Training

The model that is being used for this research is the YOLOv8 model. YOLO is an abbreviation for You Only Look Once and is an object detection model, which aims to detect and classify objects in an image or video frame. YOLOv8 is based on a deep convolutional neural network (CNN). The key idea behind YOLO is to divide the image into a grid of cells and make predictions for bounding boxes and class probabilities directly from that grid. After dividing the image into the grid cells, The CNN extracts high-level features from that image. For each cell, the model predicts multiple bounding boxes and the probability of each bounding box, belonging to a specific class. YOLOv8 then uses a non-maxima suppression (NMS) algorithm to filter out overlapping bounding boxes and select the most likely bounding boxes for each object in the image.[Tor24]
There are different YOLOv8 models available: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l and YOLOv8x. The models differ in their depth and width of their

layers. The smallest model have fewer and less complex layers compared to the biggest model, making it lighter and faster, but potentially less accurate.[Joc23]
YOLOv8 is trained using a combination of labeled bounding box annotations for the images. Then, the dataset is separated into training and validation data.

In order to get the best results out of the model (also due to the small amount of training data), a hyperparameter optimization should be conducted. In order to achieve this goal and find the best hyperparameters for this type of data, a simple grid search was implemented. The grid search has been executed, trying out different combinations of the following parameters:

- **Learning Rate:** The rate at which a model's parameters are adjusted during training, influencing the size of each step in the optimization process.
- **Batch Size:** The number of training examples processed in a single iteration, affecting the efficiency of model updates and memory usage.
- **Number of Epochs:** The total number of passes through the entire training dataset, defining the training duration and influencing convergence.
- **Network Architecture:** The structure and design of YOLO model, influencing its capacity and ability to capture complex patterns. In this research, YOLOv8x and YOLOv8m have been tested.
- **Confidence:** In the context of object detection, confidence refers to the predicted probability that an object is present, impacting the model's decision-making process.

After the first grid search was conducted and gave first insights in which parameters work best, two more grid searches have been executed to further fine tune the models performance. Each grid search took 3-4 hours of execution time.
The performance of the models has been measured using following metrics:

- **Precision:** Precision measures the accuracy of positive predictions and is calculated as the ratio of true positives to the sum of true positives and false positives.
- **Recall:** Recall (sensitivity) gauges the model's ability to capture all relevant instances and is calculated as the ratio of true positives to the sum of true positives and false negatives.
- **mAP50:** Mean Average Precision at 50 IoU measures the average precision across different confidence levels at a fixed IoU threshold of 50%. It provides an overall evaluation of precision and recall trade-offs.
- **mAP50-95:** Mean Average Precision from 50 to 95 IoU extends the evaluation to multiple IoU thresholds (50% to 95%), offering a comprehensive assessment of a model's localization performance.

2.4 Grid Search

In total there were 3 grid searches. The first one was used to find out what batchsize can be run and how many epochs will be the best, also what learning rate is the best for each architecture. It was also intended to see some differences between the architectures YOLOv8x and YOLOv8m. Additionally it was tested which impact the confidences have in the results.

2.4.1 Grid Search 1

The first grid search tried to use batch sizes of 16 and 32 but this immediately had to be changed as google colab broke down when trying to process the training for the YOLOv8x architecture with the batchsize of 32. The batchsize was then limited to 16 and a new epoch parameter was added as compensation. The Results of the grid search showed that the epochs had a huge influence on the model, the higher the epochs the better the result. It was also recognizable that a higher learning rate was better to use with the YOLOv8m model and a lower learning rate with the x model. The confidence had minor influences in the result so therefore there could not really be decided which confidence fits best, as in a few cases the model with higher confidences performed slightly better in mAP50-95 but also slightly worse in mAP50. The parameters are summarized in the table below 2

Learning Rates	0.001, 0.1
Batch Sizes	16
Epochs	6, 10
Network Architectures	YOLOv8m, YOLOv8x
Confidences	0.5, 0.75

Table 2: Grid Search Parameters 1

2.5 Grid Search 2

For the second grid search, the focus was placed on the YOLO8x model. Since the results of the first grid search indicated that higher epochs had a positive effect, it was decided to use the maximum number of epochs from the first grid search and to try out an even higher number of epochs with 16.

Learning Rates	0.001 , 0.01 , 0.1
Batch Sizes	16
Epochs	10 , 16
Network Architectures	YOLOv8x
Confidences	0.5 , 0.7 , 0.9

Table 3: Grid Search Parameters 2

2.6 Grid Search 3

After running the first two grid searches, the results were evaluated and it was decided, to use the YOLOv8m model, to compare it to the results for the YOLOv8x model from grid search 2. Grid search 2 showed, that higher epochs have a positive effect, so 16 epochs were used here. Compared to grid search 2, more values for for the learning rate and confidence were tested.

Learning Rates	0.0005 , 0.001 , 0.01 , 0.1 , 0.2
Batch Sizes	16
Epochs	16
Network Architectures	YOLOv8m
Confidences	0.5 , 0.6 , 0.7 , 0.8

Table 4: Grid Search Parameters 3

3 Results

In this section the results of the grid search 2 and 3 are evaluated and compared. As grid search 2 focused on the YOLOv8x model and grid search 3 focused on the YOLOv8m model the comparison will be made between the x and m model. For the comparison the best model of each grid search is chosen which is determined by the mAP50-95. The mAP50-95 combines the precision and recall to measure the accuracy of the object detection and localization of said model. The metric compares the actual bounding box with the predicted ones and how much the boxes overlap.

3.1 YOLOv8x Model

The best YOLOv8x model has a mAP50-95 of 0.4310 which means that on average 43% of the predicted boxes had an intersection over union (IoU) value of min 50% with the actual boxes. The model is fairly able to detect and localize the different classes in the menu.

When looking at the graph (in the Jupyter Notebook) for the development of the mAP50 and mAP50-95 one can see the development of the metrics. The graph shows that the metrics drop at the 8th epoch but after that epoch the model performs much better with the indication that even more epochs could provide even better results. While training the validation loss also rises at the 8th epoch but dramatically lowers after that.

Having a look at the real results when applied to images and comparing them with the actual boxes one could conclude that the model is not ready to be used in an application as can be seen in the notebook. But one can also see that the model is able to detect certain parts of the picture correctly and adds fitting bounding boxes which shows that the model has potential to work when evaluated further.

The model seems to be able to predict items surprisingly good, but the sections and categories are not that well predicted. This could be since there are far less sections and categories than items for example and therefore the model is not able to predict sections that well. This is also supported by the DFL loss as it is quite low, it describes that the model is able to predict the classes based on how many data was persistent in the training for each class. A low DFL loss describes that the predictions for the sections and categories are not as good because there are not that many in the training set.

To have an overview of the specifications of said model the following table is provided 5.

Learning Rate	0.001
Batch Size	16
Epoch	16
Network Architecture	YOLOv8x
Confidence	0.7

Table 5: Best YOLOv8x Model Specifications

3.1.1 YOLOv8m Model

The best YOLOv8m model has a mAP50-95 of 0.42 which means that it is very close to the performance of the YOLOv8x model.

When evaluating the curve of the mAP50 and mAP50-95 one can see that in contrast to the YOLOv8x model before, the metrics keep rising with each added epoch. For the first few epochs it is rising by large amounts but after the 7th or 8th epoch it stops increasing that dramatically. When looking at the losses it seems like they keep going down which is good. Sometimes the validation loss is increasing but after a few epochs it is nearly at the same level or even below the previous minimum.

As the mAP50-95 is almost the same as for the YOLOv8x model there should not be that much differences when looking at the predictions on real pictures. As with the YOLOv8x model the best YOLOv8m model is not that good at predicting sections and categories. This is because of the same reason as explained for the YOLOv8x model. The items are also mostly correctly predicted by the model so the models perform fairly the same.

To have an overview of the specifications of said model the following table is provided m6.

3.1.2 YOLOv8m Model 32 Epochs

As already stated in the grid search it was discovered that more epochs led to better results with validation testing. Taking a look at the best YOLOv8m model one can see that the mAP metrics are rising from epoch 15 to 16. Having that in mind the

Learning Rate	0.01
Batch Size	16
Epoch	16
Network Architecture	YOLOv8m
Confidence	0.5

Table 6: Best YOLOv8m Model Specifications

specifications of the best model regarding the mAP50-95 were taken and just the epochs were changed to 32 and a new model was trained. This model had a final mAP50-95 around 0.45 so just a bit better than the previous models.

The predictions on the actual pictures look promising so they are not too far off of the actual bounding boxes. It was decided to download two completely new pictures and let the model predict the classes. The results were better as expected, the sections of the first picture were all correct. There were a few categories overlapping but no false predictions, the only class that was not predicted very well were the items. There were a lot of overlapping items predicted but all of those predicted items were actually items so the only problem here was the overlapping.

However the predictions for the last picture were overwhelming. The model sometimes falsely predicts pictures within the menu as a category, section or item but if that is ignored the rest seems to fit how the actual bounding boxes would be.

4 Conclusion and recommendation

In conclusion, our approach to digitize restaurant menus using machine learning has shown promising results, although there are areas for improvement. The primary goal was to train a model that is able to recognize and classify different parts (or layout-sections) of a menu, including sections, category titles, and menu items. The YOLOv8 object detection model was used and extensive grid searches were conducted to optimize the hyperparameters.

There are some areas for improvements, even thought there are already promising results. The models struggled with accurate prediction of sections and category titles, likely due to the different amounts of the classes in the dataset, with fewer instances of these classes. To enhance performance, future iterations of the training should include more data to increase the diversity and quantity of data for these specific classes. Moreover, refining the model architecture and conducting more sophisticated hyperparameter tuning could lead to further improvements. The problem here will be the capabilities of colab, as batch sizes of 32 would result in errors, which could provide a better model.

There are several steps that may be taken when performing further research on this topic:

- **Increase dataset size:** As the size of the dataset used for this short-time research was very small, creating a bigger dataset would be the first reasonable step to increase accuracy and make this project production ready.
- **Add more classes:** On real menu cards there may be other important classes to recognize. E.g. ingredients of a menu item, information about additives.
- **Try other pipeline approaches:** Another idea for a pipeline solving this problem would be to create two separate models, one for detecting the sections, and a second model for detecting the items and titles within the section. Maybe this two-step approach improves the performance as the second model can focus more on the relevant parts.
- **More training:** Try more epochs with a greater batch size (as the computation costs were to high during this research)
- **Data augmentation:** Perform data augmentation on the training set to see if it can improve the model's performance by increasing the dataset size
- **Create product MVP:** Finish the pipeline to get a product MVP by adding the OCR part and creating a JSON file out of the detected text. Also include some simple (non AI) algorithm that will take the bounding box information and the detected text in it to generate that JSON file.

References

[Joc23] Glenn Jocher. What are the differences between yolov8-det and yolov8-seg of network structures, 2023. URL: https://github.com/ultralytics/ultralytics/issues/6224.

[Tor24] Jane Torres. How does yolov8 work? a peek inside its object detection brain, 2024. URL: https://yolov8.org/how-does-yolov8-work/.