Implementation and Evaluation of a MLaaS for Document Classification with Continuous Deep Learning Models

Franz Frederik Walter Viktor Walter-Tscharf

Abstract

This paper indicates an approach of a continuous training pipeline to enhance deep learning models and assessing their feasibility based on an evaluation. The purpose of this research is to analyze the quality effect of a continuously learning neural network algorithm for document classification by taking user feedback into account. The hypothesis implies that user feedback through active learning increases the precision and thus makes the process of document classification more efficient. For this purpose, based on a utility analysis, the available technologies are identified, and necessary ones are selected for designing a software concept. TensorFlow as a deep learning framework, Tesseract as an OCR engine, and Apache Airflow for the life cycle management and for orchestrating the elements for the continuous training pipeline are used. This implementation of a machine learning as a service prototype allows for exploration into the synergistic effect between the use of active learning, in the form of user feedback, and the quality of document classification achieved by deep learning. In an experiment, the implemented service is used to analyze the models behavior based on three different states. This includes synthetic data and active learning in the form of user feedback through data from data augmentation and simulated realistic data. The result shows that active learning enhanced models indicate a higher accuracy than artificially generated models. The evaluation experiment confirms the hypothesis that user feedback with continuously learning models perform better in terms of generalizing within the document classification. In conclusion, the paper demonstrates the technical requirements for implementing a machine learning as a service and affirms that the use of active learning can be integrated into existing industrial systems.

Keywords

Active learning • Document classification • TensorFlow serving • Lifecycle management • Continuous integration • Deep learning • Continuous training pipeline • Machine learning as a service • MLOps • TFX

1 Introduction

Each year, approximately 500 billion electronic invoices and documents are sent to customers or consumers worldwide (Koch, 2019, p. 12). On average, the cost of processing each business transaction is estimated at 17 US dollars (Pezza & Jan, 2012, p. 6), and is partly due to the process of capturing and digitizing or processing documents, which is still manual today. Former Chief Scientist Andrew Ng of search engine company Baidu described this process as follows: "The industrial revolution freed humanity from much repetitive physical drudgery; I now want AI to free humanity from repetitive mental drudgery, such as driving in traffic" (Ng, 2017). With this, he expresses that artificial intelligence is intended to help relieve humans of repetitive tedious work. The authors Luger and Stubblefield also define artificial intelligence as "The branch of computer science that is concerned with the automation of intelligent behavior" (Luger & Stubblefield, 1998). They reference methods and technologies within information technology, which aim to perform tasks that require intelligence in their execution (Theobald, 2019, p. 115). According to a study by McKinsey, the industries with the highest potential for automation are tourism and hospitality with 73% and manufacturing with 60% (Manyika et al., 2017, p. 7). Companies are now investing in the automation of tasks that were originally assumed to be performed only by clerks. A key issue is

F. F. W. V. Walter-Tscharf (✉)
Department of Information Engineering and Computer Science, University of Trento, Trento, Italy
e-mail: viktor.waltertscharf@unitn.it

information extraction or document recognition. Many of the use cases focus on data collection, processing, and consolidation from formats such as email, PDF, or fax (Denecken, 2018). Instead of the time-consuming manual extraction of information from documents before the transferal to an Enterprise Resource Planning–ERP–system, a clerical salesperson's work focus can be concentrated on more demanding activities through automation.

A deep learning model must be developed for recognizing key information within a document to enable generic document processing, therefore, requiring a large and annotated dataset. The downsides are addressed throughout this paper, for example in preparation, the data must be labeled, which requires time and work. This research deals with the incorporation of user feedback into deep learning models for document classification. As the concept of active learning is a new and promising topic, the goal is to continuously extend the models with user feedback through this method.

2 Literature Review

For the thematic classification of the present work, the state of the art research, related methods, and concepts of relevant scientific papers are briefly introduced and differentiated. This involves work on document recognition, extraction, and annotation. For several years, different methods have been developed focusing on research approaches for text classification, which concentrate on the interpretation of the content of a document, as well as object recognition algorithms, which identify the layout or important areas.

2.1 Theoretical Background

Chen et al. in their work on “Convolutional Neural Networks for Page Segmentation of Historical Document Images” (Chen et al., 2017) develop a page segmentation method based on a convolutional neural network—CNN. They focus the page segmentation problem on pixel labeling and propose to train features of raw image pixels through a CNN. Although the focus of their work is on recognizing handwritten documents, their approach implies that the basic idea of using a CNN to recognize important regions in PDF documents has potential. Research papers such as “Page Object Detection from PDF Document Images by Deep Structured Prediction and Supervised Clustering “(Li et al., 2018) or” ICDAR2017 Competition on Page Object Detection” (Gao et al., 2017) deal with structure and range detection of documents and also use a CNN in their approach. Their model focuses on the use of object recognition by deep learning algorithms in images of documents where—among others—tables, mathematical formulas, graphs, or figures are identified. Another approach to document classification is presented in the conference paper “Fast CNN-Based Document Layout Analysis” (Oliveira & Viana, 2017). The authors focus on document layout analysis to extract information from document images. Their model presents a one-dimensional approach for document layout analysis considering text, figures, and tables based on a CNN. They take the approach of reducing the data representativeness, of text and table blocks, to a one-dimensional CNN, which compared to the classical two-dimensional CNN approaches, should significantly improve the overall performance without affecting accuracy. This paper especially (Oliveira & Viana, 2017, Fig. 1) indicates an adequate abstract method to differentiate the structure of documents. The research paper “DeepDeSRT: Deep Learning for Detection and Structure Recognition of Tables in Document Images” (Schreiber et al., 2017) outlines a novel end-to-end system for identifying tables in images of documents through deep learning; rows, columns, and cell positions are identified. The DeepDeSRT model fundamentally consists of a CNN in conjunction with object recognition to identify the position of tables. For this, transfer learning with a pre-trained Faster R-CNN model is used.

For structure detection, DeepDeSRT uses a fully convolutional network (FCN)-based segmentation model, providing a reliable concept for document content classification. In the evaluation of the evaluation results, values between 91.44% and 96.77% are achieved for table and structure recognition.

2.2 Related Work

The topic of continuously augmenting neural networks with user feedback is sparsely covered in current literature. It was introduced and presented by Google in 2017 with the title “TFX: A TensorFlow-Based Production-Scale Machine Learning Platform” (Baylor, et al., 2017). TensorFlow Extended (TFX) is a platform developed by Google for data preparation, training, validation, and deployment of machine learning models in production environments. The platform is based on TensorFlow Core and takes as its initial idea the problem of maintaining a machine learning service. A pipeline is described consisting of different components including an analysis, transformation and validation of data, as well as a component for generating the machine learning model with the processed data. Likewise, a phase for the evaluation of the previously developed model exists, as well as a unit for the deployment of the model into the productive system. The integration of the platform and the accompanying orchestration of the components into a pipeline is

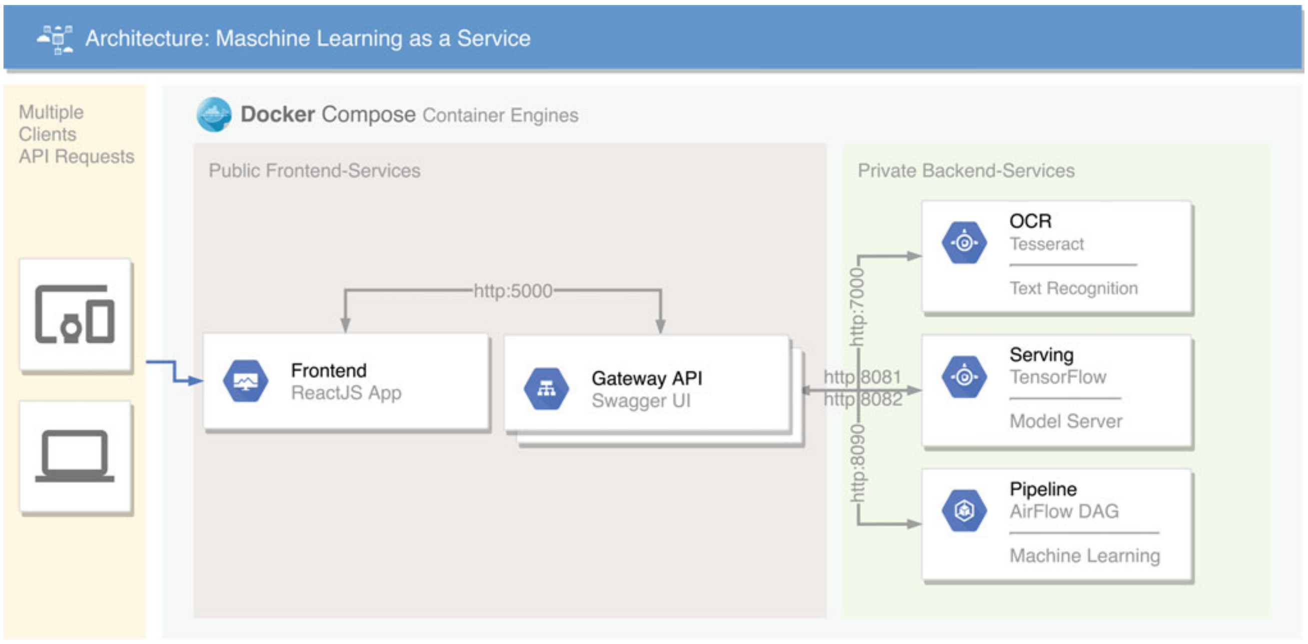


Fig. 1 Architecture of the MLaaS

intended to reduce development time and increase performance (Baylor et al., 2017, p. 1394). TFX is intended to provide an interactive platform for the user to afford deeper insight into the operation and decision path of a specific machine learning model. By building the pipeline (Baylor et al., 2017, p. 1389), the concept for a “continuously training pipeline” (Baylor et al., 2017, p. 1393) is defined.

Another scientific contribution is provided by the article “TensorFlow-Serving: Flexible and High-Performance ML Serving” (Olston et al., 2017). The authors describe an approach to use machine models productively and provide ways to feed TensorFlow with new data for training a new machine learning model. The main task of TensorFlow Serving (TFS) is to be able to productively deploy different machine learning models. RPC and HTTP interfaces are provided to interact with the model. Through TFS, execution paths for discovering new machine learning models and performing predictions of result values have been optimized to avoid performance issues of the native implementation through TensorFlow Core (Olston et al., 2017, p. 1). Routine tasks such as adding, removing, and updating a machine learning model are also handled. Additionally, rollback and canary requests, which are central to resilience, are supported. TFS is part of the end-to-end machine learning pipelines and is actively used in TFX (Olston et al., 2017, p. 6). Concepts for incorporating user feedback into the training process is provided in “Active Learning Literature Survey” (Settles, 2010). The topic is classified under the term active learning and the paper outlines its application scenarios. The implementation of these individual methods is based on the mathematical logic of (Settles, 2010, Fig. 2—1, 2, and 3). Of central importance is the “membership query” type. In this method, the artificial intelligence algorithm proposes a classification and the oracle or the user can agree or correct it (Fischer, 2000, p. 6). The scientific article “From Theories to Queries: Active Learning in Practice” (Settles, 2011) shows the practical implementation of these individual methods.

A large number of scientific papers address the topic of artificial intelligence in the area of machine learning and deep learning. The current scientific consensus is that the performance of deep learning models improves by increasing the amount of data compared to traditional machine learning algorithms. Object recognition methods based on artificial neural networks are successfully used for document classification. However, a new approach is the combination of transfer learning and active learning. By using both concepts, an autonomous self-learning system for document classification is developed utilizing user feedback.

3 Concept and Design

A clear concept and design are essential when creating a deep learning model and continuously integrating user feedback. An OCR technology which converts PDF documents into machine-readable code is necessary. Far more important, however, is a machine learning framework that can use a model to detect where critical information is located in the document. The model should be able to be improved by feedback from the user and continuously deploy new advanced deep learning models. This requires a pipeline that can process, validate, share, and transform feedback data. It must also be able to train and monitor neural networks. From these requirements for the system's functionalities, the necessary parts can be derived; an OCR engine that converts the PDF document, a serving component that provides the deep learning model and continuously waits for the latest version, and a pipeline that handles the deep learning lifecycle management. These different services are difficult to orchestrate. Figure 1 shows an overview of the described components.

The task of the serving component in the container-based microservices architecture is to make the developed model accessible via Web interfaces. For this purpose, the pipeline provides the functionalities to continuously extend the existing neural network with the user feedback from the frontend and make the new model available to the serving component. The output of the current deep learning model is processed by the OCR component after transformations have been performed in the API gateway and thus the required information is determined.

3.1 Serving

The serving component with TFS is used to address the deep learning model via Web http requests. In addition, a service is needed to continuously wait for a newly trained deep learning model in order to incorporate it at runtime when it arrives and to use it via the interfaces. Currently, only one deep learning model is needed to perform the structure determination of the document, however, TFS can manage multiple versions of the model. Through the API interface, the neural network can be integrated into the target environment of the productive system quickly, flexibly, and with high performance. For this purpose, TFS provides a REST Web API and a gRPC interface for integration. Both can be accessed independently of the programming language. For setting up TFS and the developed deep learning model, Google provides an official Docker image that includes all required resources. The drawbacks are minor; the additional software requires packages, dependencies, or libraries, which increase complexity and maintenance efforts. Moreover, an additional service can decrease the comprehensibility of the developed code, however, the effort to send an image file to a Web API endpoint would be significantly greater when implementing the interfaces natively or without TFS. In the architecture, TFS is used to address a deep learning model and identify ROIs—Region of Interests—based on images. This refers to the structure determination of the uploaded image document using object recognition in images. The return values of the ROIs are the coordinates and the designation of areas which contain important information from the document. Also returned is the precision of the particular ROI, which distinguishes between the areas Header, Content, Customer, Type, and Identification. Another aspect that would be difficult to implement without TFS is continuously waiting for new neural networks trained by the pipeline or TensorFlow. The versioning capability is intended to support jumping back to an earlier version of the model. Versioning is intended to ensure resilience and prevent significant impacts on mean average precision—mAP — which may have occurred due to the evaluation of the test datasets, the use of the MLaaS with the new deep learning model or limits to the functional capability of the service.

3.2 Pipeline

The pipeline enables the continuous processing of user feedback and thus accomplishes a constant improvement of the system. The amount of data that can be used for a new deep learning model increases with the user feedback which, along with the object classes and their coordinates, are stored in a CSV file and are then used to continuously train new versions of the model. The machine learning pipeline has eight phases, which allow for the required workflow for the training process and continuous integration (Google, 2019c). The data collection phase is responsible for preparing the collected data of the user feedback for the upcoming training process. The data preprocessing phase is used to measure and analyze the collected data. Based on the amount of data, a random determination of a dataset for evaluation and actual training is performed. In the third phase, the TensorFlow record files are created from these datasets. Phase four is used to monitor the evolution of the neural network and assess the model at runtime of the training process, based on the test dataset, using metrics for accuracy. The central utility for this purpose is TensorBoard, which is integrated in TensorFlow. The tool can be used for validating and analyzing deep learning specific parameters. Important parameters are mAP, Total Loss, Localization, Precision, and Recall. Phase five executes the training process, which includes developing a new model graph that is responsible for the prediction. The TensorFlow API provides the necessary Estimator, which represents the code for our model and manages the training process. The Estimator can be seen as a pre-built framework for training the neural network. Among other things, it also supports routine tasks like saving the learning progress of the training into so-called "check-points" or exporting the trained graph as a SavedModel, which can then be used in production environments. These steps are part of the "Save Training" and the "Export Interference Graph" phase. The final phase involves making the new deep learning model available through interaction

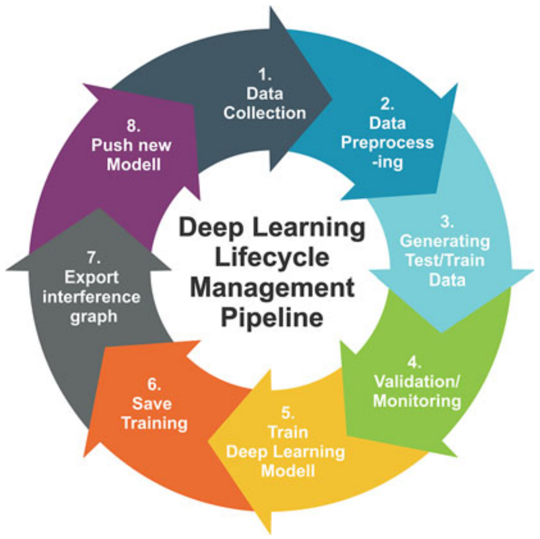

Deep Learning Lifecycle Management Pipeline
1. Data Collection
2. Data Preprocess -ing
3. Generating Test/Train Data
4. Validation/ Monitoring
5. Train Deep Learning Modell
6. Save Training
7. Export interference graph
8. Push new Modell

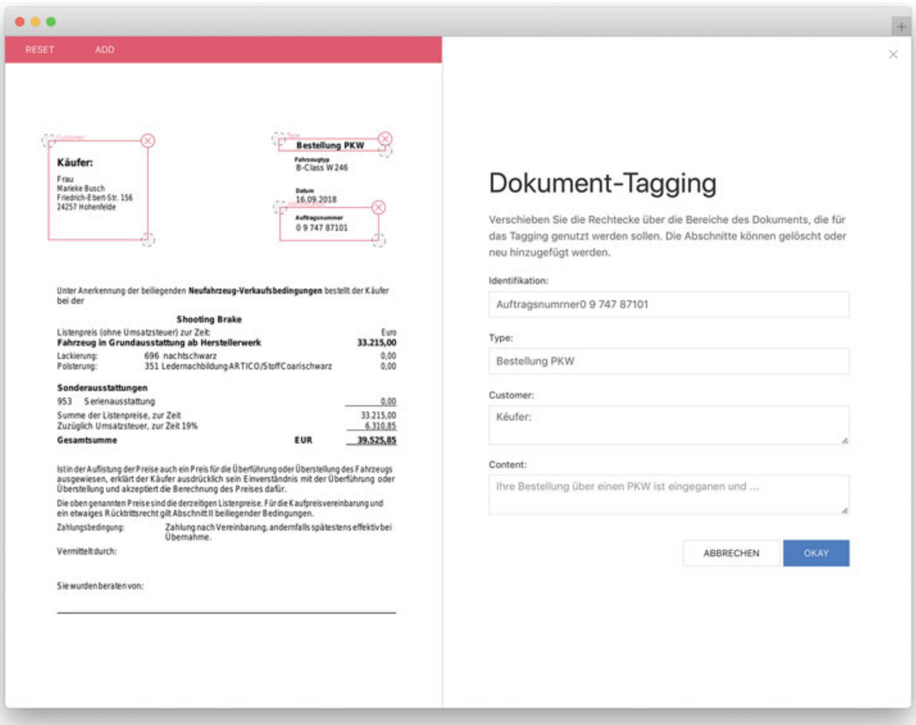
Fig. 2 Deep learning lifecycle management pipeline

with the serving component. The model is initiated dynamically without interrupting the service at runtime. After all phases of the pipeline have been executed, the result is a new deep learning model available at the serving component which has been augmented with user feedback from the previous queries. An overview and abstract framework of the flow is described in Fig. 2.

The phases must be combined to form an overall structure, the result of which is the Continuous Training Pipeline. This monitors the correct progress of the individual phases in a targeted manner takes over the orchestration or arrangement of the phases and merges them into an overall construct. Google recommends Apache AirFlow (Google, 2019a) for these requirements. In Airflow, the combined flow or pipeline is called a DAG— directed acyclic graph. Here, the DAG consists of a collection of all the tasks required to fulfill the workflow (Apache Airflow, 2020). In practice, DAGs are Python files, which in turn contain multiple operators describing a single task in a workflow. In the described pipeline, the operators represent the different phases and, using AirFlow, the logs and code of these can be viewed individually. In addition, Airflow provides interfaces to trigger the modeled pipeline or the DAG via Web requests. The REST API can also be used to activate/deactivate the DAG, query the status, and retrieve execution time or task information. These functionalities are only indirectly accessible from the client. The gateway component manages this REST API and corresponding requests are only forwarded if required. This approach is chosen to comply with the architecture guidelines and the SoC.

The Web application is developed with the frameworks ReactJS and UIKit as demonstrated in Fig. 3. The user has the possibility to define the content of specific tags by moving the rectangles and thus providing corrections which form the necessary feedback. For the implementation of the animation and graphical functionality, the D3.JS framework

Fig. 3 Adding feedback to the suggested tags of a document in form of content areas, movable as rectangles



Käufer:

Frau
Marieke Busch
Friedrich-Ebert-Str. 156
24257 Hohenfelde

Bestellung PKW

Fahrzeugtyp
B-Class W246

Datum
16.09.2018

Auftragsnummer
0 9 747 87101

Unter Anerkennung der beiliegenden **Neufahrzeug-Verkaufsbedingungen** bestellt der Käufer bei der

Shooting Brake

Listenpreis (ohne Umsatzsteuer) zur Zeit:		Euro
Fahrzeug in Grundausstattung ab Herstellerwerk		**33.215,00**
Lackierung:	696 nachtschwarz	0,00
Polsterung:	351 Ledernachbildung ARTICO/Stoff Caorischwarz	0,00
Sonderausstattungen		
953 Serienausstattung		0,00
Summe der Listenpreise, zur Zeit		33.215,00
Zuzüglich Umsatzsteuer, zur Zeit 19%		6.310,85
Gesamtsumme	**EUR**	**39.525,85**

Ist in der Auflistung der Preise auch ein Preis für die Überführung oder Überstellung des Fahrzeugs ausgewiesen, erklärt der Käufer ausdrücklich sein Einverständnis mit der Überführung oder Überstellung und akzeptiert die Berechnung des Preises dafür.

Die oben genannten Preise sind die derzeitigen Listenpreise. Für die Kaufpreisvereinbarung und ein etwaiges Rücktrittsrecht gilt Abschnitt II beiliegender Bedingungen.

Zahlungsbedingung: Zahlung nach Vereinbarung, andernfalls spätestens effektiv bei Übernahme.

Vermittelt durch:

Sie wurden beraten von:

Dokument-Tagging

Verschieben Sie die Rechtecke über die Bereiche des Dokuments, die für das Tagging genutzt werden sollen. Die Abschnitte können gelöscht oder neu hinzugefügt werden.

Identifikation:
Auftragsnummer0 9 747 87101

Type:
Bestellung PKW

Customer:
Käufer:

Content:
Ihre Bestellung über einen PKW ist eingeganen und ...

is used and communication with the backend utilizes the JS Fetch API. The gateway is implemented using FlaskREST-Plus. The serving component in the container-based microservices architecture allows the developed model to be accessible via Web interfaces and is implemented using TensorFlow Serving. The pipeline continuously augments the existing neural network with user feedback and provides the new model to the Serving component. The deep learning model will be developed using TensorFlow and for the orchestration, AirFlow is used. A translation of the images into text is achieved with the OCR component by Tesseract and Flask. In combining these subcomponents, a document classification service through a continuous deep learning training pipeline is outlined and available (Walter-Tscharf, 2021).

4 Evaluation

An assessment and evaluation of the benefits of user feedback as well as the continuous expansion of the deep learning models are carried out via an experiment with the developed prototype. The basis of the assessment is a model that is continuously extended through active learning and is available in a total of three different states, which compared using an evaluation dataset.

- Model 1: Faster R-CNN and synthetic data
- Model 2: Model 1 and active learning in the form of data augmentation
- Model 3: Model 2 and active learning in the form of data simulation

The Faster R-CNN model acts as the foundation.

4.1 Metric

The methodology behind the assessment follows the approach of the Pascal Challenge (Everingham et al., 2010, pp. 313–314). The research paper “A Comparative Analysis of Object Detection Metrics with a Companion OpenSource Toolkit” (Padilla, 2019) shows an applicable approach for evaluating the deep learning models. The focuses are the metrics Accuracy, Precision, Recall, Loss, and the $F1$-score demonstrated in Eqs. 1–6.

$$\begin{aligned}\text{IOU} &= \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area}\left(B_p \cap B_{gl}\right)}{\text{area}\left(B_p \cup B_{gt}\right)} \\ &= \frac{\left|B_p \cap B_{gr}\right|}{\left|B_p \cup B_{gt}\right|}\begin{cases}\text{TP, IOU} \geq \text{ threshold} \\ \text{FP, IOU} < \text{ threshold} \\ \text{FN}, B_{gl} = \emptyset\end{cases}\end{aligned} \quad (1)$$

$$\begin{aligned}\text{accuracy} &= \frac{\text{true positives } + \text{ true negatives}}{\text{all samples}} \\ &= \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}\end{aligned} \quad (2)$$

$$\text{precision} = \frac{\text{true positives}}{\text{all detections}} = \frac{\text{TP}}{\text{TP } + \text{ FP}} \quad (3)$$

$$\text{recall} = \frac{\text{true positives}}{\text{all ground truths}} = \frac{\text{TP}}{\text{TP}+\text{FN}} \quad (4)$$

$$\begin{aligned}\text{mAP} &= \frac{1}{N}\sum_{j=1}^{n} AP(j) \\ \text{AP} &= \frac{1}{M}\sum_{j=1}^{N} \text{precision}(k)\end{aligned} \quad (5)$$

$$F_1 \text{ - score } = 2 \cdot \frac{\text{precision } \cdot \text{ recall}}{\text{precsion } + \text{ recall}} \quad (6)$$

In practice, these values were determined by combining the Toolkit and the TensorBoard.

4.2 Experiment

Model 1 is developed using only synthetic data and transfer learning. Models 2 and 3 represent a use of the developed prototype to directly extract user feedback from the document classification system to extend the neural network. Model 2 is intended to assess what impact user feedback with data augmentation has on document classification. Model 3 will be used to investigate the impact of using data with simulated real-world scenarios.

The different training patterns are shown in Figs. 4, 5, 6, 7, 8, and 9.

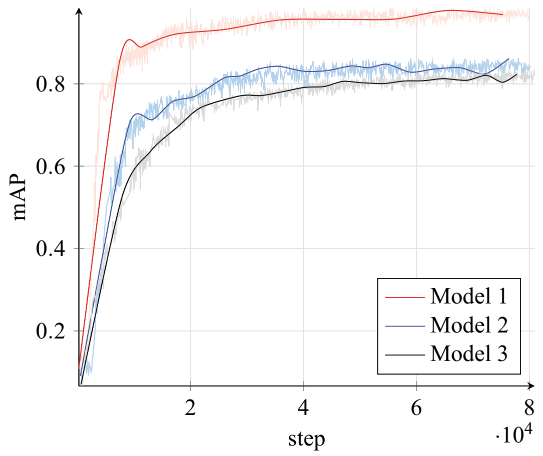


Fig. 4 Detection boxes precision/mAP of experiments

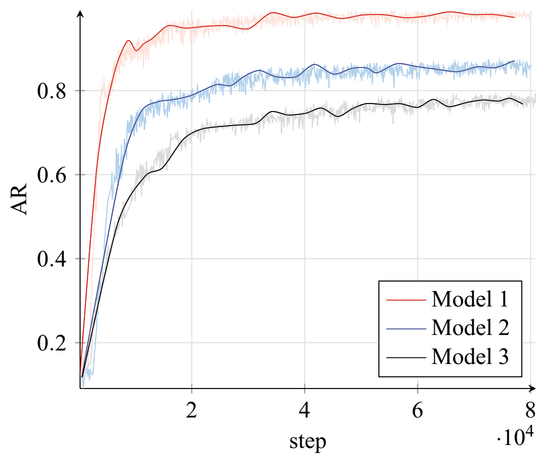


Fig. 5 Detection boxes recall/AR of experiments

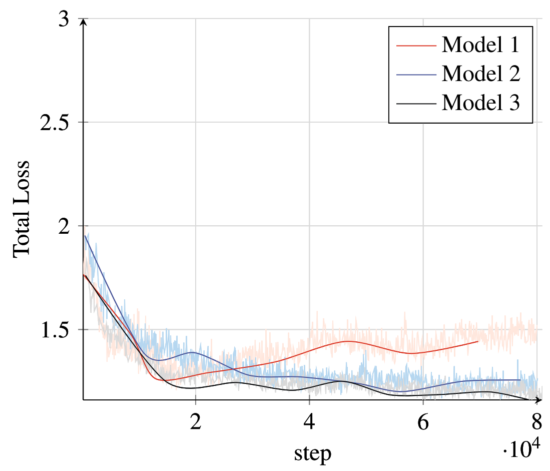


Fig. 8 Total loss of experiments

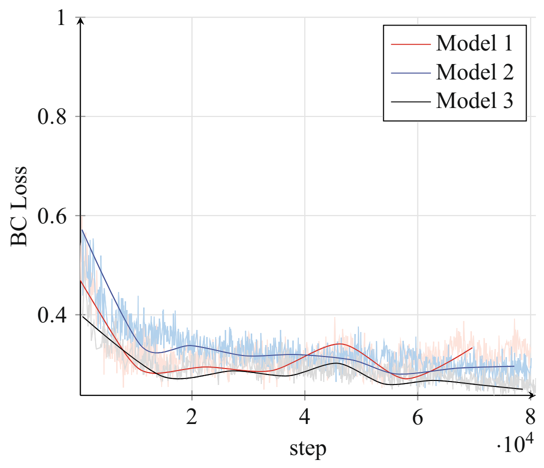


Fig. 6 BC Loss/localization loss of experiments

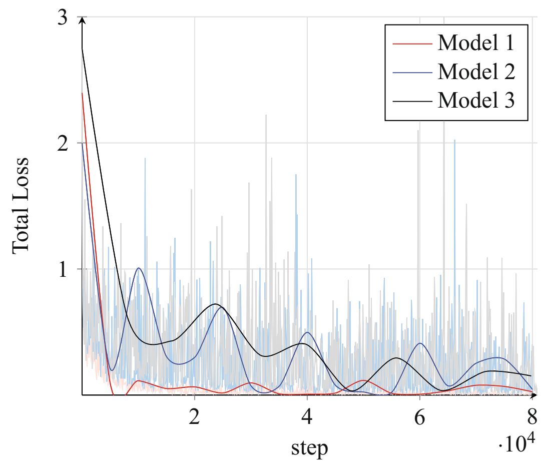


Fig. 9 Train total loss of experiments

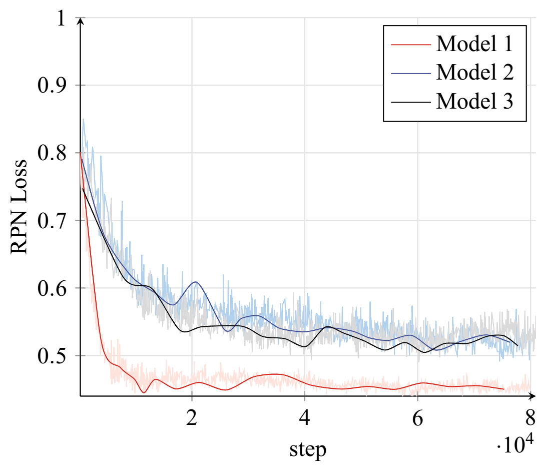


Fig. 7 RPN Loss/localization loss of experiments

After the performed training of the models, the three states of the model exhibit the characteristic values given in Table 1.

It is remarkable that for Models 2 and 3, which both use user feedback, the mAP decreases. Further analysis is required into the behavior of the models when an evaluation dataset from a real-world scenario is present, which the models are not trained for. That is to say, if the generalization for Models 2 and 3 increased. In addition, the new evaluation dataset also includes new types of documents that were not previously considered. Relevant to this is the use of documents without content or arbitrary documents and without object classes or labels. Each of the mentioned models is run with the evaluation dataset to test the hypothesis of the impact of user feedback. Central to this is the analysis of the different states of the models using the metrics, see Table 2.

Table 1 Overview of the results from the three models in the different stages

Model	Iteration	Recall (AR)	Precision (mAP)	$F1$-score	Total loss
Model 1	82,532	0.9777	0.9693	0.9735	1.4017
Model 2	80,150	0.8617	0.8451	0.8533	1.2648
Model 3	80,925	0.7852	0.8221	0.8032	1.1916

Table 2 Evaluation of the different models

Model	Iteration	Recall (AR)	Precision (mAP)	$F1$-score	Total loss
Model 1	82,532	0.9777	0.9693	0.9735	1.4017
Model 2	80,150	0.8617	0.8451	0.8533	1.2648
Model 3	80,925	0.7852	0.8221	0.8032	1.1916

5 Results

Discussing the results, a distinction must be made between training the models and running the experiment with an evaluation dataset. Training the models assesses whether the transfer learning is performed correctly, whereas the evaluation assesses the quality in terms of the respective generalizability of the three models as a result of an experiment with the same dataset— the evaluation dataset. Figures 4, 5, 6, 7, 8, and 9. show the metrics relevant for the training process; mAP, Recall/AR, BC Loss, RPN Loss and Total Loss refer to the test dataset, and train Total Loss refers to the training dataset. For model 1, the mAP increases steeply up to the 10,000– 15,000 iteration line and then converges to a value between 0.95 and 0.98. Models 2 and 3 have a similar trend; the mAP level increases steeply up to the value 15,000 and converges to a range of values from 0.81 to 0.87 and 0.78 to 0.82, respectively. This indicates a lack of diversity or the greater homogeneity of the training dataset in Model 1 compared to Models 2 and 3. AR and mAP are similar in the course and with respect to their values. It can be inferred that FN and FP have similar values. The values of mAP and AR of 0.96–0.80 from Table 1 conclude that only low values for FN and FP occur. All values of the Failure or Loss functions are absolute values and do not represent percentages. The value of the Box Classifier Localization Loss of all models varies between 0.15 and 0.35, implying that all models are at a similar level with respect to the different test datasets and the region of interest detected. Here, the BC Loss of the localization refers to the last layer of the Faster R-CNN model, which is responsible for determining the position of a bounding box (Lee et al., 2019, p. 1). The RPN Loss of the models is in a comparable range; the converging values are at 0.45 for model 1 and 0.52 for Models 2 and 3. The low first value could be due to the small horizontal and vertical displacement of the data. The Total Loss of the three models is within the range of 1.2 and 1.5 after the 20,000 iteration. The low variation could be on account of the minimum of a loss function not being found among the summed Total Loss. To evaluate the training process, the Total Loss of the training dataset is described in addition. A low value here corresponds to a model with a higher quality— unless the model tends to overfit the training data (Arsalan Soltani & Chen, 2015). Each of the models has a value below 0.2, therefore, the loss functions of the training should be sufficiently minimized. Crucial for the final assessment of the training quality of the models is the $F1$-score, presented in Table 1; the values range from 0.97 for model 1, 0.85 for model 2, and 0.80 for model 3 for a comparable number of iterations. The predictive ability of model 1 result values is the highest, however, this is accompanied by an increasing Total Loss— 1.40 for model 1 compared to 1.26 for model 2 and 1.19 for model 3. This means that while the accuracy of the first model increases compared to Models 2 and 3, the misses also increase. This indicates increasing generalizability of Model 2 and 3 compared to Model 1. The values of the three models show little change with further iterations. The improvement of the mAP and AR values are minimal. The values of the loss functions are minimized or have a small increasing tendency.

Therefore, the models are sufficiently trained (Srivastava et al., 2014, p. 1). Another training would not increase the performance, resulting in an overfitting of the models. The requirements for the models to perform an evaluation are thus fulfilled.

Results of the evaluation experiment are given in Table 2. The $F1$-scores for the three models when tested with the same evaluation dataset (Table 1) are lower than the $F1$-scores of the test dataset in training (Table 2). The reason for this is the larger variance of the data of the evaluation dataset compared to the test dataset of the training. It is noticeable that contrary to the results of the training, the $F1$-score in the evaluation experiment increases from model 1 with 0.6, to model 2 with 0.65, to model 3 with 0.72. This is evidence that the predictive ability of the models trained using active learning is greater than the first model trained using only synthetic data Model 3, which was trained based on data augmentation and data simulations using active learning, achieves the highest results. The lower total loss of Model 1 (2.23), Model 2 (1.55), and Model 3 (1.19) also illustrates increasing quality by incorporating user feedback.

6 Discussion and Limitations

The experiments and results confirm that active learning improves the quality of models in terms of diversification and applied generalization. It also reduces the effort of the manual labeling of data. The results of increasing accuracy of document recognition by the developed MLaaS show that systems which learn and continuously improve with active learning through user feedback have a higher probability of achieving a desired prediction of results than alternative systems. The result of the evaluation is the behavior of a deep learning model studied in three different states. These are created by continuous training with different data; the basis is a model based on synthetic data, and the other two models—for investigating the continuous augmentation of deep learning models—are implemented by incorporating user feedback. The data used is generated using data augmentation and a simulation of real data. The result of the training is that the $F1$-score for the models with active learning decreases due to the homogeneity between training and test datasets and the different structure of the datasets. The evaluation experiment confirms the hypothesis that the continuously extended models have a better generalizability with regard to document recognition.

The limitations and opportunities of the present prototype naturally span across different domains. Enhancements in the area of text classification, a model metadata database, or the development of an outsourced service for autonomous model validation are considerable. The scope of the current implementation does not include functionalities for user authentication. The service OCR, serving, and pipeline are not able to simply adapt authentication methods— such as a "bearer token". A limitation occurs due to the storage of user feedback in a CSV list, which compromises the ACID principle (atomicity, consistency, isolation, and permanence or persistence), which is supposed to ensure transaction security. Therefore, a database like SQLite or PostgreSQL would be useful for the metadata of the model. Google has introduced the MLMD—ML Metadata— concept for this purpose, which makes all relevant information for the model accessible (Google, 2010b). Another significant limitation is associated to the belief that the user will provide qualified valuable feedback. The pipeline would continue to produce new models, however, the models would be redundant, missing an improved increase in the precision An interesting approach to solve this would be by taking a psychological factor into account, for instance a reward system for the user. Even though these limitations exist, the developed prototype can simply be transferred to other related use cases without significant challenges. Apart from the defined object classes or label names, all approaches presented in the concept are free of context-specific algorithms. Essentially, they are defined by the data used for training. If a comparable dataset was available for a similar use case, the overall system could be transferred with little effort.

7 Conclusion

The objective of this paper is to develop a generalized scientific concept as a basis for the further development of deep learning models using user feedback. The working hypothesis is the assumption that a software learns via user feedback and thus improves the system's efficiency. The topic of artificial intelligence in relation to machine learning and deep learning is covered increasingly extensively in the current research literature. Deep learning is well suited for object recognition tasks, as the performance of the models and the algorithms improve as the size of the data increases. Relevant error tolerances for the training process of deep learning models are overfitting and underfitting. The paper shows that object recognition methods based on CNNs can be successfully used for document classification. In order to create the possibility to continuously extend a deep learning model, a pipeline and a lifecycle management are necessary. The combination of transfer learning and active learning is novel in this context. It is used to develop an autonomous and self-learning system for document classification based on user feedback data. The practical implementation of single partial solutions is already presented in the literature: however, the combination as a complete system exists only as a theoretical approach.

The software solution developed in this paper is able to capture documents and independently determine the content relevant to the business process. The user can supplement the data determined and suggested by the system as needed and furthermore has the possibility to adjust the data. The development concept of a prototype requires two basic technologies; an OCR engine and a machine learning framework to identify content areas using the deep learning model. TensorFlow as a deep learning framework and Tesseract for text recognition are selected. The container-based microservice architecture consists of the components frontend, middleware (API gateway), and backend services (Serving, Pipeline and OCR). For the frontend, React is provided as the Web framework and UIkit and D3 for the design. The API gateway in the software solution is based on Python in combination with Flask for the required interfaces. The API is designed and documented in the software solution using a Swagger UI. Serving, as a component of the backend service, allows the deep learning algorithm to be accessible via the Web. A pipeline allows for the existing neural network to be continuously extended with user feedback from the frontend and provides the serving component with the new model. Apache Airflow is used for lifecycle management and the accompanying orchestration of the elements of the pipeline. The OCR microservice uses Tesseract to translate images into text. As a result, the implementation of the software concept has created a prototype that is used to classify documents, and the system is continuously being extended as part of active learning with the Faster R-CNN deep learning model. The quality of the deep learning models can be technically monitored and compared using the TensorBoard. Metrics for validation include Mean Average Precision, Average Recall, Total Loss, and $F1$-score.

Via the frontend, a user can easily move the recommendation of the recognized content area—visualized by a rectangle in the image of the document—and thus correct the classification of the uploaded document. Challenges in the implementation are the conversion of relations with regard to the selection mask, the displayed images of the document, and the original sizes of the document.

The subject of the evaluation is an experiment with a deep learning model using the newly developed machine learning as a service. This uses active learning with user feedback from the document classification system directly, thereby continuously expanding the neural network. The behavior of the model is analyzed based on three states which are created by continuous training with different data. The first state is a model based on synthetic data and the other two models are implemented by using the service and considering user feedback—for the second state with data from data augmentation and for the third state additionally with real data generated in a simulation. The experiment consists of both training and experimental components, which gather opposite results. The training shows that the $F1$-score for the models with active learning decreases. This is caused by the homogeneity between training and test datasets and the different structure of the datasets for each model. However, in the experiment with a test dataset of potentially simulated realistic images, the recognition quality of the models increases when using active learning. Hence, a deep learning model trained with only synthetic data performs worse in evaluation and practice than enhanced models trained with user feedback through active learning. The evaluation experiment, therefore, confirms the hypothesis that the continuously augmented models have improved generalizability and a continuous training pipeline increases accuracy with respect to document recognition.

Acknowledgements The research for this paper was financially supported by Exprivia S.p.A, University of Trento (UniTrento), Fondazione Bruno Kessler (FBK) and the European Institute of Innovation & Technology (EIT).

References

Apache Airflow. (2020). *Airflow documentation concepts*. Retrieved from Airflow Documentation Concepts: https://airflow.apache.org/concepts.html

Arsalan Soltani, A., & Chen, K. (2015). *Neural network—How to interpret loss and accuracy for a machine learning model*. Retrieved from Neural network – How to interpret loss and accuracy for a machine learning model: https://stackoverflow.com/questions/34518656/how-to-interpret-loss-and-accuracy-for-a-machine-learning-model

Baylor, D., Koc, L., Koo, C. Y., Lew, L., Mewald, C., Modi, A. N., Jain, V. (2017). *TFX: A tensorFlow-based production-scale machine learning platform*. Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD '17, pp 1387–1395. ACM Press. https://doi.org/10.1145/3097983.3098021

Chen, K., Seuret, M., Hennebert, J., & Ingold, R. (2017). *Convolutional neural networks for page segmentation of historical document images*. 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), pp. 965–970. IEEE. doi: https://doi.org/10.1109/ICDAR.2017.161

Denecken, S. (2018). *SAP intelligent robotic process automation—Adding the "Brain" and "Skillsets" to your digital workers*. Retrieved from SAP Intelligent Robotic Process Automation—adding the "Brain" and "Skillsets" to your digital workers: https://blogs.sap.com/2018/09/17/sap-intelligent-robotic-process-automation-adding-the-brain-and-skillsets-to-your-digital-workers/

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). *The Pascal Visual Object Classes (VOC) challenge*, pp. 303–338. Retrieved from https://doi.org/10.1007/s11263-009-0275-4

Fischer, P. (2000). *Algorithmisches Lernen*. Teubner.

Gao, L., Yi, X., Jiang, Z., Hao, L., & Tang, Z. (2017). *ICDAR2017 competition on page object detection*. 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), pp. 1417–1422. IEEE. https://doi.org/10.1109/ICDAR.2017.231

Google. (2019a). *Orchestrating TFX pipelines*. Retrieved from Orchestrating TFX Pipelines: https://www.tensorflow.org/tfx/guide/airflow

Google. (2019b). *ML metadata TFX*. Retrieved from ML Metadata TFX: https://www.tensorflow.org/tfx/guide/mlmd

Google. (2019c). *The TFX user guide*. Retrieved from The TFX User Guide: https://www.tensorflow.org/tfx/guide

Koch, B. (2019). *International market overview & forecast*, p. 23.

Lee, S., Kwak, S., & Cho, M. (2019). *Universal bounding box regression and its applications*. Retrieved from http://arxiv.org/abs/1904.06805

Li, X.-H., Yin, F., & Liu, C.-L. (2018, August). *Page object detection from PDF document images by deep structured prediction and supervised clustering*. 2018 24th International Conference on Pattern Recognition (ICPR), pp. 3627–3632. IEEE. doi:https://doi.org/10.1109/ICPR.2018.8546073

Luger, G. F., & Stubblefield, W. A. (1998). *Artificial intelligence: structures and strategies for complex problem solving* (3rd ed ed.). Addison-Wesley.

Manyika, J., Chui, M., Miremadi, M., Bughin, J., George, K., Willmott, P. J., & Dewhurst, M. (2017*). A future that works: automation, employment and productivity*.

Ng, A. (2017). *Opening a new chapter of my work in AI*. Retrieved from Opening a new chapter of my work in AI: https://medium.com/andrewng/opening-a-new-chapter-of-my-work-in-ai-c6a4d1595d7b

Oliveira, D. A., & Viana, M. P. (2017). *Fast CNN-based document layout analysis*. 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), pp. 1173–1180. IEEE. doi:https://doi.org/10.1109/ICCVW.2017.142

Olston, C., Fiedel, N., & Gorovoy, K. (2017). *TensorFlow-serving: Flexible, high-performance ML serving*, 2, 8. Retrieved from http://learningsys.org/nips17/assets/papers/paper_1.pdf

Padilla, R. (2019). *Object-detection-metrics*. Retrieved from Object-Detection-Metrics: https://github.com/rafaelpadilla/Object-Detection-Metrics

Pezza, S., & Jan, W. (2012). AP invoice management in a networked economy. *AP Invoice Management in a Networked Economy*, 25.

Schreiber, S., Agne, S., Wolf, I., Dengel, A., & Ahmed, S. (2017). DeepDeSRT: Deep learning for detection and structure recognition of tables in document images. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1162–1167. IEEE. https://doi.org/10.1109/ICDAR.2017.192

Settles, B. (2010). *Active learning literature survey*, p. 67.

Settles, B. (2011). *From theories to queries: Active learning in practice*, p. 18.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). *Dropout: A simple way to prevent neural networks from overfitting*, p. 30.

Theobald, E. (2019). *Marketing Intelligence: ein Lehrbuch für die Praxis* (1. Auflage ed.). Verlag W. Kohlhammer.

Walter-Tscharf, V. (2021). *Source code implementation and evaluation of a MLaaS for document classification with continuous deep learning models*. Retrieved from Source code implementation and evaluation of a MLaaS for document classification with continuous deep learning models: https://github.com/FranzTscharf/ml-pipeline