Ostfalia
Hochschule für angewandte Wissenschaften

Fakultät Informatik

PowerPointDoktor: Toolbox zum Transformieren und Optimieren von PowerPoint-Präsentationen

Jan Braun
Matrikel-Nr. 70474566

Bachelorarbeit im Studiengang Informatik
zur Erlangung des akademischen Grades:
Bachelor of Science

Ostfalia Hochschule für angewandte Wissenschaften

1. Prüfer: Prof. Dr.-Ing. habil. Dirk Joachim Lehmann
2. Prüfer: Prof. Tobias Dörnbach, Ph.D.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere, dass ich alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe, und dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist.

Braunschweig, 18.11.2023

Ort, Datum

Unterschrift

Zusammenfassung

Präsentationen sind heutzutage ein weit verbreitetes Medium zur Vermittlung von Informationen. Nicht immer liegen diese Präsentationen in verschiedenen Sprachen vor und müssen daher, im besten Fall automatisch, übersetzt werden. Diese Arbeit befasst sich mit der Herausforderung, die mit der Übersetzung von Texten in Präsentationen verbunden ist, insbesondere in den Bildern, die in den Präsentationen gezeigt werden. Durch den Einsatz von Bilderkennungswerkzeugen, die mit Hilfe von neuronalen Netzen in der Lage sind Texte zu erkennen und zu extrahieren, soll eine Anwendung geschaffen werden, die diese Herausforderung bewältigt.

In dieser Arbeit wird ein Konzept erarbeitet sowie der PowerPointDoktor umgesetzt. Diese Anwendung ermöglicht die Übersetzung von PowerPoint-Präsentationen und wird darüber hinaus als Toolbox zur Transformation und Optimierung von PowerPoint-Präsentationen mittels verschiedener Funktionen zur Verfügung gestellt.

Zur Evaluierung des PowerPointDoktors werden Aspekte wie die Übersetzungszeit, die Qualität der übersetzten Textbilder und ein detailliertes Praxisbeispiel behandelt. Die anschließende Diskussion und der Ausblick geben eine Bewertung der Evaluation, stellen die aufgetretenen Probleme dar und fassen die Arbeit sowie mögliche Ansätze zur Verbesserung der Anwendung zusammen.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ANN	Artificial Neural Network
API	Application Programming Interface
BGR	Blau, Grün, Rot Farbraum
CNN	Convolutional Neural Network
CRAFT	Character Region Awareness For Text Detection
CRNN	Convolutional Recurrent Neural Network
FCN	Fully Convolutional Network
GNMT	Google Neural Machine Translation
GUI	Graphical User Interface
HTTPS	Hypertext Transfer Protocol Secure (englisch für "sicheres Hypertext-Übertragungsprotokoll“)
JSON	JavaScript Object Notation
KE	Kreuzentropie
MAE	Mean Absolute Error (englisch für "mittlere absolute Fehler")
MSE	Mean Squared Error (englisch für "mittlerer quadratischer Fehler")
NMT	Neural Machine Translation
ONMT	Open Neural Machine Translation
QML	Qt Modeling Language
ReLU	Recitified Linear Unit
RGB	Rot, Grün, Blau Farbraum
RNN	Recurrent Neural Network

UI	User Interface
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language

1. Einleitung

Die Motivation und Ziele der Bachelorarbeit werden in diesem Kapitel erläutert. Außerdem wird die Struktur der Arbeit vorgestellt und eine thematische Abgrenzung vorgenommen.

1.1. Motivation

Die Thematik des Übersetzens von Texten in andere Sprachen hat in den letzten Jahren vermehrt an Bedeutung und Interesse gewonnen. [56] So gibt es Informationen in verschiedenen Formen und Medien, die es zu vermehren gilt. Die Fähigkeit, Inhalte in verschiedene Sprachen zu übertragen, ermöglicht es Menschen auf der ganzen Welt miteinander zu kommunizieren. In diesem Kontext hat sich die Neural Machine Translation (NMT) als ein starkes Werkzeug erwiesen, um schnell und vor allem genaue Übersetzungen zu generieren.

Mit Übersetzen ist jedoch nicht nur die Übersetzung von Texten gemeint. Grund dafür ist unter anderem der Fortschritt in der Bildverarbeitungstechnologie, mit welcher Texte in Bildern erkannt werden können. Dies eröffnet eine neue Möglichkeit der Informationsverarbeitung, denn oft enthalten Bilder oder Grafiken für den Leser relevante Informationen. Interessant ist hierbei der Anwendungsfall bei PowerPoint-Präsentationen, welche weltweit in den unterschiedlichsten Bereichen als Kommunikationsmittel eingesetzt werden. So nutzen laut einer Studie des Marktforschungsinstitutes INNOFACT, bei der 1.022 Personen zur Nutzung von Programmen zur Erstellung von Präsentationen befragt wurden, nur 3% PowerPoint überhaupt nicht und mindestens 17% PowerPoint täglich. [23] Die Umfrage zeigt weiterhin, dass die alternativen Programme Prezi und Keynote von den Befragten nur zu 30% überhaupt genutzt werden. Betrachtet man zudem die Nutzung pro Woche, so bleiben nur 16,6% für Keynote und 12,6% für Prezi übrig. Dies zeigt, dass PowerPoint zu den am häufigsten genutzten Werkzeugen zur Erstellung von Präsentationen gehört. [24]

Die PowerPoint-Präsentationen liegen nicht immer in einer Sprache vor, die von allen Zuhörern verstanden wird, und ihre Übersetzung erweist sich in der Regel als zu zeitaufwändig und ressourcenintensiv. Außerdem neigen Menschen dazu Fehler zu machen und sind sich dessen nicht immer bewusst. Beim menschlichen Übersetzen könnte ein fehlender oder lückenhafter Wortschatz des Übersetzers ein Problem darstellen. Gerade in der Lehre oder im Berufsleben werden viele PowerPoint-Präsentationen erstellt und verteilt. Bei einem internationalen Publikum und Kunden aus dem Ausland ist ein Programm zur Unterstützung bei der Übersetzung dieser Präsentationen hilfreich. Missverständnisse, die aufgrund von unverständlichen oder fehlerhaften Übersetzungen zustande kommen, können somit vermieden werden.

1.2. Zielsetzung der Arbeit

Ziel dieser Bachelorarbeit ist die Entwicklung eines Programms, das in der Lage ist, den Inhalt einer PowerPoint-Präsentation in verschiedene Sprachen zu übersetzen. Dabei wird sowohl der geschriebene Text als auch der in den Bildern erkannte Text berücksichtigt. Obwohl die automatische Texterkennung fehlerhaft sein kann, bringt sie dennoch ein erhebliches Zeitersparnis mit sich, die zugunsten menschlicher Nachbearbeitung genutzt werden kann. Darüber hinaus wird so eine inklusivere und barrierefreiere Kommunikation mit einem internationalen Publikum gefördert.

Neben der Übersetzungsfunktion soll das Programm weitere nützliche Funktionen bieten, die es dem Nutzer erleichtern wird, eine Vielzahl an PowerPoint-Präsentationen schnell und zentral zu verwalten. Diese zusätzlichen Funktionen tragen dazu bei, den Arbeitsablauf bei der Erstellung und Verwaltung von Präsentationen zu optimieren und ermöglichen es dem Nutzer, sich auf den inhaltlichen Aspekt zu konzentrieren.

1.3. Gliederung der Arbeit

In Kapitel 2 werden die Grundlagen von neuronalen Netzwerken erklärt und spezialisierte neuronale Netzwerke, die sich mit der Texterkennung in Bildern und der Übersetzung von Sprachen beschäftigen, betrachtet. Kapitel 3 behandelt das Design der Applikation, definiert die Benutzeroberfläche und beschreibt die erforderlichen Funktionen. Mit der Umsetzung der Anforderungen beschäftigt sich Kapitel 4, wobei die Bildübersetzungslogik näher erläutert wird und das für die Oberflächenentwicklung verwendete Framework vorgestellt wird. In den folgenden Kapiteln 5 und 6 werden die Evaluierung und die Beurteilungen zur Überprüfung der Übersetzungsqualität vorgenommen. Abschließend wird in Kapitel 7 ein Ausblick auf mögliche Weiterentwicklungen und Verbesserungen der Implementierung gegeben.

1.4. Thematische Abgrenzung

Eine fehlerfreie Übersetzung von Texten in Bildern wird in dieser Arbeit nicht möglich sein. Grund dafür sind verschiedene Faktoren, die die Texterkennung einschränken: Dazu gehören unter anderem komplexe Hintergründe, das Vorhandensein komplexer Schriftarten, -größen und -ausrichtungen. [22, Seite 2]

2. Grundlagen

Das folgende Kapitel beschreibt die theoretischen Grundlagen dieser Bachelorarbeit. Zuerst wird die Struktur eines *Artificial Neural Network (ANN)* vorgestellt und anschließend genauer auf die Spezialform des ANN, dem *Convolutional Neural Network (CNN)* eingegangen. Diese finden Nutzen in der Identifizierung von Textbereichen in Bildern. Im Anschluss wird die Grundlage zum Übersetzen der erkannten Textbereiche und der daraus resultierenden Literale erläutert.

2.1. Grundsätze von neuronalen Netzwerke

Neuronale Netzwerke haben in den vergangen Jahren vermehrt an Interesse gewonnen. [55] Um diese zu verstehen muss der grundlegende Baustein, das Neuron, verstanden werden.

Ein biologisches Neuron bekommt Informationen in Form von Signalen von verschiedenen anderen biologischen Neuronen. Diese Informationen haben weiterhin eine unterschiedliche Gewichtung. Sollte das biologische Neuron durch den Eingang der Informationen eine Stimulation erfahren, so wechselt es in einen anderen Zustand und gibt ein Signal weiter, falls dies nicht der Fall sein sollte, so verbleibt es in seinem ursprünglichen inaktiven Zustand. [2] Ähnlich hierzu verhält es sich mit einem artefaktischen Neuron. Abbildung 1 stellt beispielhaft einen Aufbau eines solchen Neurons dar.

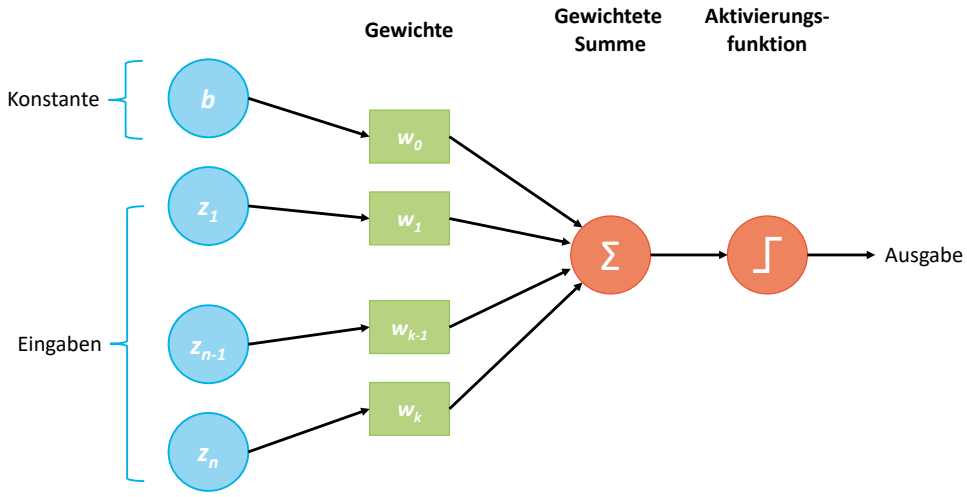


Abbildung 1.: Darstellung eines artefaktischen Neurons. (Basierend auf: [49])

Es ist möglich ein artefaktisches Neuron auch als folgende Funktion $f : \mathbb{R}^k \to \mathbb{R}$ darzustellen, diese wird aus der linearen Abbildung $q : \mathbb{R}^k \to \mathbb{R}$ und einer nichtlinearen Funktion, welche auch Aktivierungsfunktion genannt wird, $g : \mathbb{R} \to \mathbb{R}$ aufgebaut. Zusammengesetzt bedeutet dies

$$f(z_1, ..., z_k) = g(q(z_1, ..., z_k)) \tag{2.1}$$

wobei $z_1, ..., z_k$ die Ausgabe der Neuronen der vorhergehenden Schicht ist. [17, Seite 26]. Die lineare Abbildung, die die gewichtete Summe der Ausgaben der vorherigen Schicht darstellt, wird wie folgt mathematisch beschrieben:

$$q(z_1, ..., z_k) = \sum_{j=1}^{k} w_j z_j + b \tag{2.2}$$

In diesem Kontext gibt es die reellen Zahlen $w_1, ..., w_k$, welche *Gewichte* genannt werden, diese spiegeln die Stärke der Bindung zu den Neuronen der vorherigen Schicht wider. Wohingegen $b \in \mathbb{R}$ als *Bias* bezeichnet wird, welcher die Aktivierungsschwelle des Neurons durch einen konstanten Versatz beeinflusst.

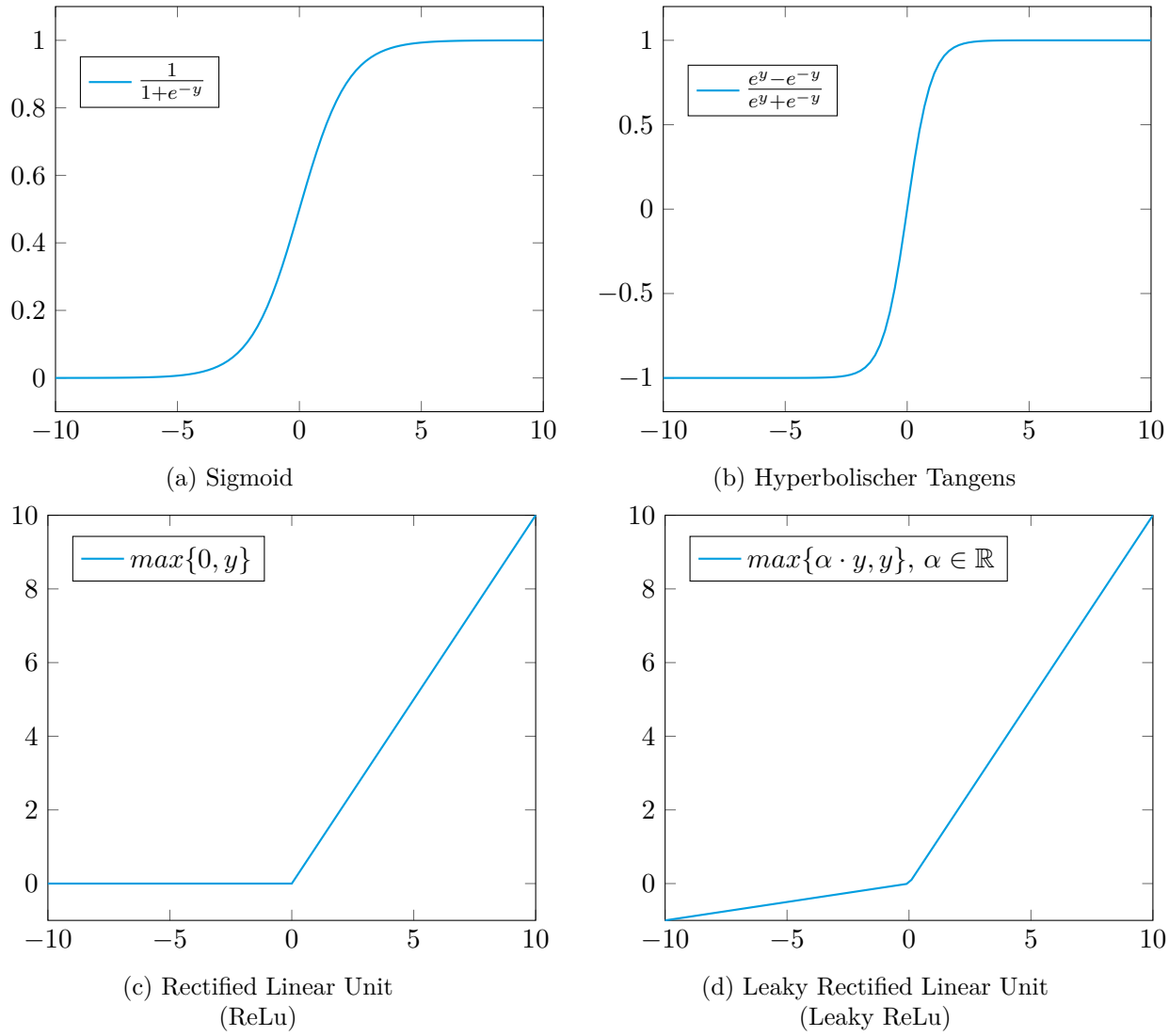
Die Auslösung eines Neuronen mittels Aktivierungsfunktion(en)

Das Anwenden der Aktivierungsfunktion g ist wichtig, um eine Transformation zwischen den Ein- und Ausgaben zu erzielen. Ohne die Anwendung dieser Funktion käme die Nutzung eines neuronalen Netzes lediglich linearen Transformationen gleich. Mit den Aktivierungsfunktionen ist es erst möglich für ein neuronales Netzwerk aus den Informationen zu lernen, diese zu reflektieren und sich gegebenfalls anzupassen. Somit wird es möglich jeden erdenkliche Prozess als eine funktionale Berechnung darzustellen. Deshalb wird eine Aktivierungsfunktion eingesetzen, um das Netz dynamisch zu machen und ihm die Fähigkeit zu verleihen, auch komplexe Informationen aus Daten zu extrahieren und nichtlineare, zufällige funktionale Zuordnungen zwischen Eingabe und Ausgabe darzustellen. Eine weitere wichtige Eigenschaft einer Aktivierungsfunktion ist, dass sie differenzierbar sein muss, damit wird es möglich Fehler und Verluste in Bezug auf die Gewichte anzupassen und diese somit zu reduzieren. [50, Seite 311]

Die Sigmoid Funktion Gleichung 2.3, dargestellt in Abbildung 2a, ist eine verbreitete Aktivierungsfunktion. Sie bildet die Eingabewerte auf Ausgabewerte zwischen 0 und 1 ab. Somit ist sie sinnvoll für *binary classification* und *logistic regression* Probleme. Sie wird meist in Fully Convolutional Network (FCN) verwendet, weil sie es dem Netz ermöglicht, Nichtlinearität in das Modell einzuführen, wodurch das neuronale Netz komplexere Entscheidungsgrenzen erlernen kann. Als Beispiel in dem Themenfeld der Bilderkennung kann die Sigmoid Funktion genutzt werden, um die Ausgabe des linearen Modells zu einer Wahrscheinlichkeit umzuwandeln. Diese Wahrscheinlichkeit kann eine Widerspiegelung hinsichtlich des Klassifizierungsproblems sein.

Sie spielt daher eine wichtige Rolle bei der Entscheidung, ob die Ausgabe binär ist oder ob die Wahrscheinlichkeit eines Ereignisses geklärt werden muss.

$$f(y)_{sigmoid} = \frac{1}{1 + e^{-y}} \tag{2.3}$$



(a) Sigmoid

(b) Hyperbolischer Tangens

(c) Rectified Linear Unit (ReLu)

(d) Leaky Rectified Linear Unit (Leaky ReLu)

Abbildung 2.: Übersicht üblicher Aktivierungsfunktionen. (Basierend auf: [17, Seite 28])

Abbildung 2b zeigt die hyperbolische Tangens-Funktion. Diese verhält sich ähnlich der Sigmoidfunktion, mit dem Unterschied, dass sie ursprungssymmetrisch ist. Außerdem ist sie stetig und differenzierbar, da die Werte im Bereich von -1 bis 1 liegen. Verglichen mit der Sigmoidfunktion

ist der Gradient der hyperbolische Tangens-Funktion steiler.

$$f(y)_{tanh} = \frac{e^y - e^{-y}}{e^y + e^{-y}} \tag{2.4}$$

Die Recitified Linear Unit (ReLU) ist eine weit verbreitete nichtlineare Aktivierungsfunktion in neuronalen Netzwerken. Sie ist in Abbildung 2c dargestellt. Da nicht alle Neuronen gleichzeitig aktiviert werden, sondern nur eine bestimmte Anzahl von Neuronen gleichzeitig, ist ReLU effizienter als andere Funktionen. In einigen Fällen ist der Wert des Gradienten gleich null, so dass die Gewichte und Verzerrungen während des Backpropagation-Schrittes nicht aktualisiert werden, wenn das neuronale Netz trainiert wird.

$$f(y)_{ReLU} = max\{0, y\} \tag{2.5}$$

Bei der in der Abbildung 2d dargestellten Leaky ReLU Aktivierungsfunktion handelt es sich um eine verbesserte Version der ReLU Funktion. Dabei wird der Wert der ReLU-Funktion für negative Werte von x nicht als null, sondern als extrem kleine lineare Komponente von x definiert. [50, Seite 313] [4, Seite 18] Mathematisch wird dies folgendermaßen ausgedrückt:

$$f(y)_{LeakyReLU} = max\{\alpha * y, y\}, \alpha \in \mathbb{R} \tag{2.6}$$

Die Softmax-Funktion schätzt das *Posterior* und wird für die Ausgabeschicht verwendet. Sie wird häufig für Mehrklassenklassifikationsprobleme und auch für binäre Klassifikation verwendet. Da die Exponentialfunktion normiert ist, ist die Summe des gesamten Vektors gleich 1. Daher kann die Ausgabe der Softmax-Funktion als die Wahrscheinlichkeiten interpretiert werden, dass ein bestimmter Satz von Merkmalen einer bestimmten Ausgabe von K entspricht. Aus diesem Grund ist dies eine sehr beliebte Softmax-Funktion, die für die Aufgabe der Mehrklassenklassifikation verwendet wird. [50, Seite 314]

$$softmax(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}} \tag{2.7}$$

Schichten innerhalb eines neuronalen Netzwerkes

Layer, im deutschen Schichten genannt, sind ein grundlegender Baustein der Architektur eines neuronalen Netzwerks. Abbildung 3 zeigt einen beispielhaften Aufbau eines neuronalen Netzwerks mit seinen drei typischen Schichten.

Die Schichten bestehen aus mehreren Neuronen, welche zur Klassifizierung und Vorhersage von Daten verwendet werden. Es gibt am Anfang eine Eingabeschicht und am Ende eine Ausgabeschicht, dazwischen befinden sich eine oder mehrere versteckte Schichten.

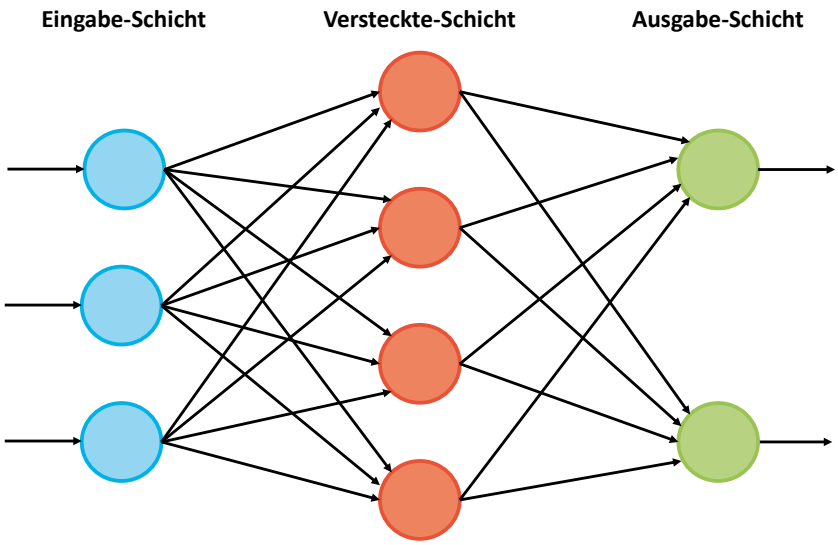


Abbildung 3.: Grafische Darstellung eines artefaktischen neuronalen Netzwerkes. (Basierend auf: [1, Seite 902])

Lediglich die erste und die letzte Schicht sind direkt zugänglich und werden daher auch als sichtbare Schichten bezeichnet. Wohingegen die versteckten Schichten weder durch eine Eingabe bestimmt sind, noch eine direkte Ausgabe geben. [17, Seite 28] Alle Schichten haben Knoten und jeder Knoten hat eine Gewichtung, die bei der Verarbeitung von Informationen von einer Schicht zur nächsten berücksichtigt wird. [50, Seite 311]

Lernmechanismen eines neuronalen Netzes

Ein wichtiger Punkt bei der Verwendung von neuronalen Netzen ist das Training des Netzes, denn durch die Modellierung ist das Netz noch nicht in der Lage die Eingaben mit einer exakten Abdeckung auf die Ausgabe abzubilden. Dazu muss das Netz erst einmal wissen, was eine korrekte Ausgabe wäre. Das bedeutet, dass zunächst definiert werden muss, welche Eingabedaten für ein Training verwendet werden können. Dabei ist zu beachten, dass die Größe des Datensatzes auch zu einer höheren Genauigkeit des Netzes führen kann. [44, Seite 2] Beim Lernen kann zwischen überwachten und unüberwachten Lernverfahren unterschieden werden. Bei dem überwachten Lernverfahren sind die zur Verfügung stehenden Daten bereits mit einem Label versehen. [26, Seite 4] Durch das Benutzen von gekennzeichneten Ein- und Ausgaben kann das Modell die Genauigkeit besser im Überblick haben und dadurch gegebenenfalls Anpassungen am Modell durchführen und somit lernen. Darüber hinaus existiert das unüberwachte Lernen mit

unmarkierten Daten. Hier wird es dem Modell selbst überlassen, Zusammenhänge und Muster in den Daten zu erkennen. [17, Seite 47] Überwachtes Lernen wird vor allem in der Bilderkennung eingesetzt. Im Gegensatz zum unüberwachten Lernen muss hier jedoch ein Aufwand in die Beschaffung und Kennzeichnung der Daten investiert werden, während beim unüberwachten Lernen menschliche Arbeit in die Nachbearbeitung und Kontrolle der Resultate investiert werden muss. Unüberwachtes Lernen kann im Vergleich zu überwachtem Lernen leistungsfähiger sein, weil es komplexe Muster erkennt, die mit anderen Methoden schwer aufzuspüren sind. Einer der Gründe dafür ist, dass Menschen auf der Grundlage ihres Vorwissens voreingenommene Annahmen treffen.

Für das überwachte Lernen kann ein vorhandener Datensatz in drei Teilmengen unterteilt werden, die sich nicht überlappen dürfen. Die Untergruppen sind Trainingsdaten, Validierungsdaten und Testdaten. Es ist wichtig, dass diese Daten so nah wie möglich an den zukünftigen realen Daten sind. Eine mögliche Aufteilung könnte zum Beispiel 70:15:15 sein. [17, Seite 21] Die Trainingsdaten, auf denen das Modell trainiert wird, machen 70% aus, die Validierungsdaten, die der Risikoschätzung des trainierten Modells dienen, 15%. Die restlichen 15% sind Testdaten, die der abschließenden Bewertung des Modells dienen. [26, Seite 128]

Aus den vorangegangenen Kapiteln ist die Existenz von Gewichten bekannt, mit denen das Netz parametrisiert werden kann. Die Gewichte eines Netzes werden in der Regel mit zufälligen Werten initialisiert, die aus einer Standardverteilung ausgewählt werden. [21, Seite 31] Das ANN verwendet die Trainingsdaten multipliziert mit dem Gewicht der Kante, um das Ergebnis in den Knoten der verborgenen Schicht zu speichern. Das Netz muss diesen Prozess für jede Trainingsdateneingabe, jede Kante, jeden Knoten und jede verborgene, sowie sichtbare Ausgabeschicht durchlaufen. Auf diese Ausgabe wird dann die Aktivierungsfunktion angewendet um die eigentliche Ausgabe zu erzeugen. Dieser Prozess wird als *Forward Propagation* oder auch Vorwärtspropagation bezeichnet. Nach jeder Iteration werden die Validierungsdaten für eine Bewertung verwendet. Mit Hilfe der Verlustfunktion wird die Fehlerquote zwischen den prognostizierten und den tatsächlichen Ausgaben berechnet. Diese Werte werden verwendet, um mit Hilfe des *Backward Propagation*-Algorithmus die Gewichte rückwärts von der Ausgabeschicht zur Eingabeschicht anzupassen, um die Fehlerquote zu reduzieren. [17, Seite 33]

Für diese Berechnung kann beispielsweise der Mean Squared Error (englisch für "mittlerer quadratischer Fehler") (MSE) genutzt werden:

$$MSE = \frac{1}{n} \sum_{j=1}^{n} e_j^2 \tag{2.8}$$

Wobei $e_j = (y_j - \hat{y}_j)$ mit y_j für die gewünschte Ausgabe des neuronalen Netzwerkes und $\hat{y}_j$ die tatsächliche Ausgabe des neuronalen Netzwerkes. [12, Seite 32]

Eine weitere Verlustfunktion, die verwendet werden kann, ist der Mean Absolute Error (englisch für "mittlere absolute Fehler") (MAE), welcher auch wie der MSE bei Regressionsproblemen verwendet wird. [12, Seite 30] Diese Funktion fasst die absolute Differenz zwischen der gewünschten und der tatsächlichen Ausgabe des Netzwerkes zusammen. Mathematisch wird dies wie folgt dargestellt:

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|e_j| \tag{2.9}$$

Die Kreuzentropie (KE) Verlustfunktion wird bei der binären Klassifikation verwendet. Sie misst, wie gut ein Klassifikationsmodell mit Ausgangswerten im Bereich von 0 bis 1 abschneidet. Bei Abweichungen der Vorhersage vom Zielwert nimmt der Kreuzentropieverlust zu. [64, Seite 2] Sie wird wie folgt definiert:

$$KE = -\sum_{j=1}^{n}(y_j log(\hat{y}_j)) \tag{2.10}$$

Nachdem mit Hilfe der Verlustfunktionen der Fehler berechnet wurde, werden die Gewichte angepasst, sodass der Fehler kleiner wird. Dieser Lernprozess wird solange auf die Trainingsdaten angewendet bis sich die Gewichte kaum verändern. Einen wesentlichen Einfluss auf den Trainingserfolg hat die Lerngeschwindigkeit. Bei zu geringer Lerngeschwindigkeit ist die Suche von vornherein auf das möglicherweise ungünstigste Minimum fixiert, hingegen verhindert eine zu hohe Lerngeschwindigkeit die Konvergenz vollständig. Aus diesem Grund wird stattdessen eine dynamische Lernrate gewählt, die reduziert werden kann, wenn die Validierungsdaten keine signifikante Verbesserung der Ergebnisse zeigen.

Zur Erzeugung eines robusten und zuverlässigen Netzes werden den Trainingsdaten in einigen Fällen Rauschen oder andere Zufallsfaktoren hinzugefügt, um das Netz mit dem Rauschen und den natürlichen Schwankungen der realen Daten vertraut zu machen. Schlechte Trainingsdaten führen unweigerlich zu einem unzuverlässigen und unvorhersagbaren Netz. In der Regel wird das Netz für eine vorgegebene Anzahl von Iterationen oder bei Unterschreitung einer bestimmten Fehlerschwelle durch den Anfangsfehler trainiert. Es ist besonders darauf zu achten, dass das Netz nicht übertrainiert wird. Durch das Übertrainieren kann sich das Netz zu sehr an die Proben aus dem Trainingssatz gewöhnen und ist dann möglicherweise nicht mehr in der Lage, Proben außerhalb des Trainingssatzes genau zu klassifizieren. [1, Seite 904]

2.2. Identifizierung von Textbereichen in Bildern

Neuronale Netze werden in verschiedenen Anwendungsgebieten eingesetzt, unter anderem bei der Identifizierung von Objekten in Bildern [5], dem visuellen Tracking [59], Bilder Klassifizierung [63] oder semantischen Segmentierung [34] - allgemein gesprochen der Bildererkennung. Hierfür eignet sich vor allem das CNN, dieses wird in dem folgenden Kapitel thematisiert.

2.2.1. Convolutional Neural Network

Im Vergleich zu den traditionellen ANNs sind die CNNs im Hinblick auf die selbstlernenden und damit selbstoptimierenden Eigenschaften der ANNs ähnlich aufgebaut. Der Hauptunterschied zwischen diesen Netzwerken ist, dass CNNs primär im Kontext der Bildererkennung und Erkennung von Mustern genutzt werden. Somit ist es möglich bilderspezifische Merkmale in die Netzwerkarchtitektur von CNNs einzubauen und gleichzeitig die Parameter für das Aufsetzen des Modells gering zu halten. [35, Seite 3] Das Netz wird so trainiert, dass es anfangs Muster wie Linien oder Kurven erkennt. Diese Informationen können anschließend verwendet werden, um komplexere Muster wie Gesichter oder Objekte zu identifizieren. [57, Seite 5]

Möglich wird dies unter Nutzung der drei spezifischen Schichtenarten des CNNs. Es gibt die *Convolutional*-Schicht, die *Pooling*-Schicht und die *Fully-Connected*-Schicht. Die ersten beiden Schichten werden genutzt, um *Features* aus dem *Input* zu identifizieren, wohingegen die letzte Schicht die Klassifizierung durchführt und somit die Ausgabe auf eine bestimmte Klasse zuordnet. Abbildung 4 zeigt beispielhaft, wie die Schichtenarten verwendet werden, um eine Zahl zu klassifizieren.

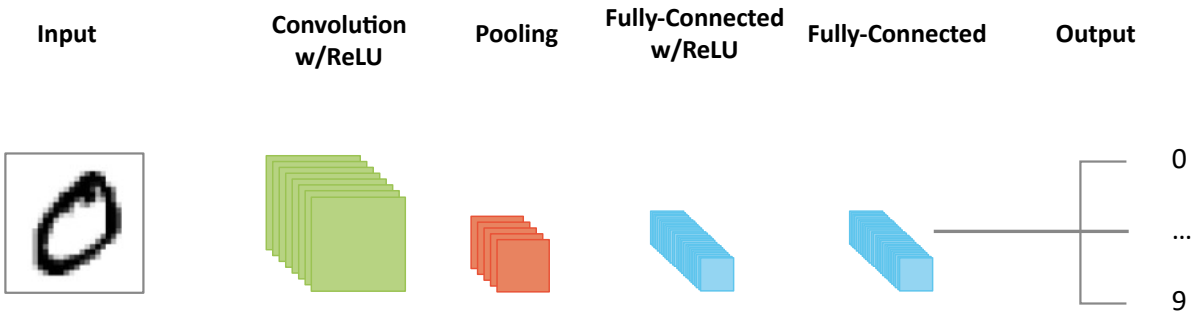


Abbildung 4.: Beispiel Darstellung eines CNNs zur Erkennung von Zahlen. (Basierend auf: [35, Seite 4])

Convolution ist eine spezielle Art der linearen Operation zur Merkmalsextraktion, bei der eine Matrix aus Zahlen, der sogenannte Kernel, auf die Eingabe angwendet wird. Diese Eingabe ist ein Array von Zahlen, das als Tensor bezeichnet wird. Ein elementweises Produkt zwischen jedem Element des Kernels und dem Eingabetensor wird an jeder Stelle des Tensors berechnet und

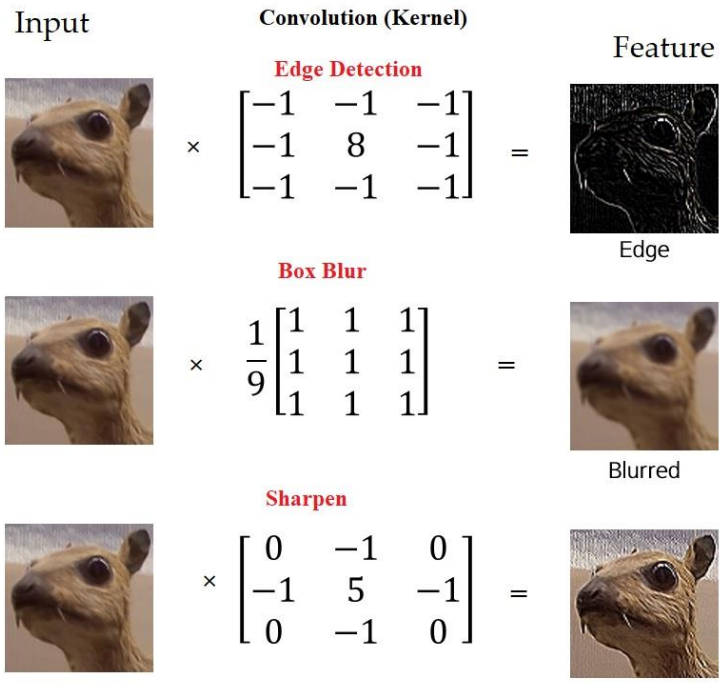


Abbildung 5.: Extraktion bestimmter Merkmale mittels Kernel-Anwendung. [57, Seite 7]

summiert, um den Ausgabewert an der entsprechenden Stelle des Ausgabetensors zu erhalten, der als *Feature Map* bezeichnet wird. [62, Seite 613] Es existieren verschiedene Kernel, welche bestimmte Merkmale aus Bildern hervorheben können. Abbildung 5 zeigt beispielhaft die Wahl eines Kernels und dem daraus resultierenden Ausgabetensors. Diese Werte sollten jedoch nicht manuell vorgegeben sein. Das Netzwerk erlernt die Kernelwerte selbst, um so die Merkmale bestmöglich eigenständig zu extrahieren. Ein Vorteil beim Benutzen von Kernels ist es, dass die Werte des Kernels gelernt werden müssen und somit ein *Weight Sharing* stattfindet, weil das Netzwerk keine neuen Gewichte zwischen den Neuronen lernen muss. Dies erhöht die Effizienz des Netzwerkes, weil hiermit Zeit und Ressourcen gespart werden. [57, Seite 7] [62, Seite 614]

Abbildung 6 zeigt, wie der Kernel schrittweise auf den Eingabetensor angewendet wird, um die *Feature Map* zu erhalten. In der Grafik findet das Verschieben des Kernels jeweils um eine Position statt. Diese Verschiebung zwischen zwei aufeinanderfolgenden Kernelpositionen wird als *Stride* bezeichnet. Meist wird die Distanz von eins gewählt, jedoch sind auch Distanzen größer eins möglich. Bei solchen Größen wird ein *Downsampling* erzielt, was einer *Pooling*-Operation gleich kommt. [62, Seite 614]

Beim Benutzen des Kernels wird deutlich, dass so gut wieder jeder Bereich mit seinen Informationen überlappt. Dies trifft jedoch nicht für den Rand zu, denn hier gehen Informationen verloren, welche jedoch mittels *Padding* wieder eingebunden werden können. Hierfür wird der Eingabetensor am Rand mit Nullen aufgefüllt. Die Anzahl an weiteren Reihen und Spalten kann

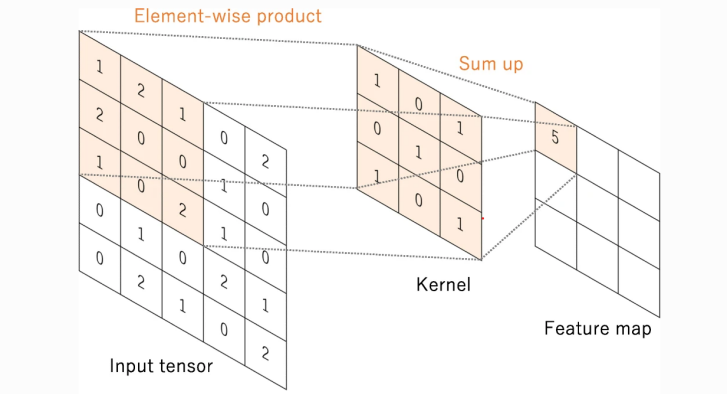


Abbildung 6.: *Convolutional*-Operation angewandt auf einen Eingabetensor. [62, Seite 614]

unter Betrachtung der Dimensionen des Kernels gewählt werden. [57, Seite 8]

Neben der *Convolutional*-Schicht existiert noch die *Pooling*-Schicht, deren Hauptaufgabe das Zusammenfassen der *Feature Map* zu kleineren *Feature Maps* ist. Gleichzeitig werden die dominanten Informationen in jedem Schritt beibehalten. Ähnlich wie bei der *Convolutional*-Schicht können hier *Stride* sowie Kernel-Größen festgelegt werden. Zu den bekanntesten und am häufigsten genutzten *Pooling*-Methoden gehören das *Max-*, *Min-* und *Global-Average-Pooling*, diese werden beispielhaft in Abbildung 7 dargestellt. [4, Seite 17] Beim Zusammenführen der *Feature Maps* wird die Lage der Daten nicht beibehalten. [57, Seite 9] Außerdem gibt es in keinem der *Pooling*-Schichten einen lernfähigen Parameter. [62, Seite 615]

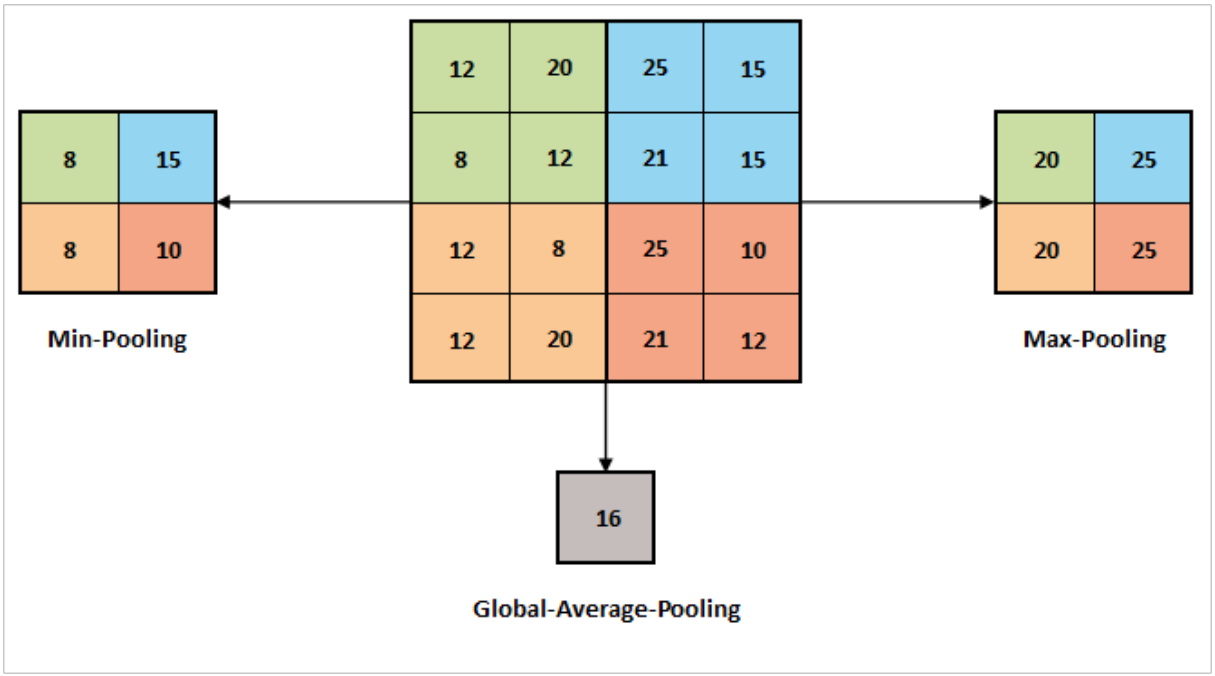


Abbildung 7.: Darstellung von drei *Pooling*-Methoden. (Basierend auf: [57, Seite 10])

Die letzte Schicht ist häufig eine *Fully-Connected*-Schicht. In dieser Schicht wird die Ausgabe mit Hilfe aller Features der letzten *Pooling*-Schicht berechnet und in einen 1D-Vektor umgewandelt, der mit den *Fully-Connected*-Schichten verknüpft wird. In dieser Schicht ist jeder Eingang mit jedem Ausgang über ein Gewicht verbunden. Die Wahrscheinlichkeiten für jede Klasse werden als Endergebnis des Netzes ausgegeben. Die Anzahl der Ausgabeknoten entspricht in der Regel der Anzahl der Klassen. [62, Seite 616]

2.2.2. Fully Convolutional Network

Eine weitere ANN Spezialisierung ist das FCN, welches dafür genutzt werden kann semantische Segmentierungsprobleme zu lösen, Bilder Klassifizierung zu erzielen oder Objekt Detektion durchzuführen. Bei der semantischen Segmentierung wird jeder Pixel eines Eingabebildes einer bestimmten Klasse zugeordnet. Die Architektur, dargestellt in Abbildung 8, eines FCN besteht aus einer Abfolge von *Convolutional*-Schichten, *Pooling*-Schichten und *Deconvolutional*-Schichten, somit ist es im Grunde ein CNN, bei dem die *Fully-Connected*-Schichten durch *Convolutional*-Schichten ersetzt wurden. Durch das *Upsampling* werden die ursprünglich kleinen *Feature Maps* auf eine höhere Auflösung skaliert. [51, Seite 640] Diese *Convolutional*-Schichten

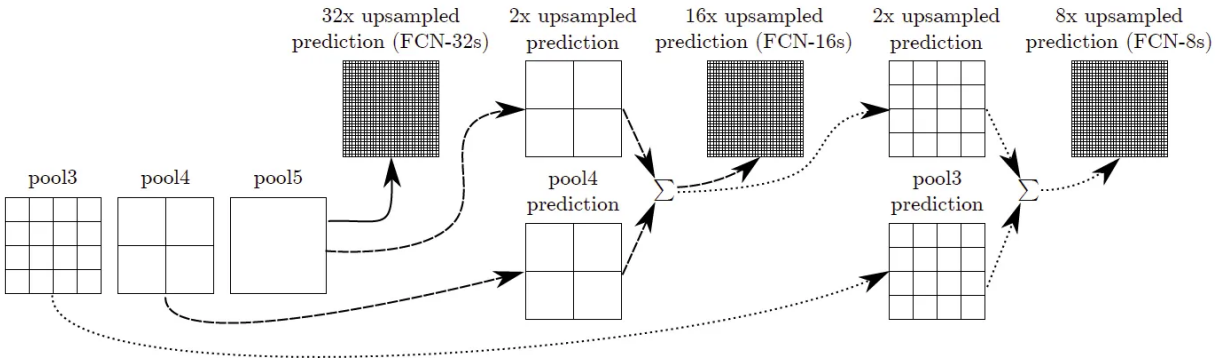


Abbildung 8.: Darstellung eines *Fully Convolutional Networks* mit seinen Schichten. [27, Seite 6]

wenden eine Operation an, welche *Transposed Convolutions* oder auch *Fractionally Strided* genannt wird. Die Eingabe in eine *Transposed-Convolutional*-Schicht ist das Ergebnis einer direkten Convolutional-Operation, die auf eine ursprüngliche *Feature Map* angewendet wurde. [18, Seite 21] Der Vorteil hierbei ist, dass die Gewichte bei *Transposed-Convolution*-Operationen auch trainiert werden können. Ähnlich zum *Subsampling* können hier verschiedene *Upsampling*-Methoden angewendet werden. Abbildung 9 zeigt beispielhaft eine dieser Methoden. Hier wird der *Input* mittels *Zero-Padding* vergrößert und im Anschluss der Kernel, dargestellt in grau, auf diesen angewendet, um die Ausgabe zu erzeugen.

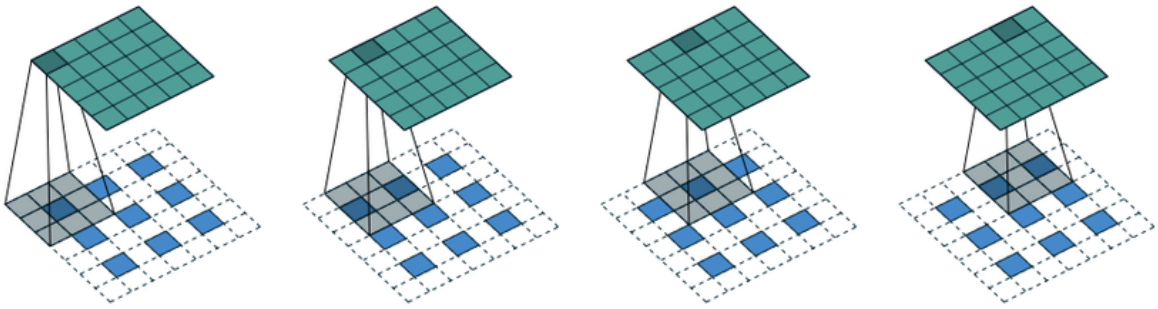


Abbildung 9.: *Upsampling* mittels *Zero-Padding*. [18, Seite 25]

2.3. Von identifiziertem Text zu Literalen

Mit Hilfe neuronaler Netze können auch Texte aus Bildern extrahiert und interpretiert werden. Auf diese Weise kann eine automatische Transkription beziehungsweise Übersetzung ermöglicht werden. Weiterhin ist eine allgemeine Informationsgewinnung und Inhaltsanalyse möglich. In den folgenden Kapiteln werden die Grundlagen hierfür betrachtet.

2.3.1. Convolutional Recurrent Neural Network

Ein Convolutional Recurrent Neural Network (CRNN) ist eine Art neuronales Netz mit einer Kombination aus *Convolutional*- und *Recurrent*-Schichten, sowie einer *Transcription*-Schicht. Bei der Eingabe von Daten in ein CRNN werden die Informationen wie folgt verarbeitet: Die Informationen werden mittels *Convolutional*-Schichten extrahiert und in *Feature Maps* übergeben. Auf dieser Schicht bauen die *Recurrent*-Schichten auf, welche anhand der Features Vorhersagen generieren. Die letzte Schicht, die *Transcription*-Schicht, überführt die Vorhersagen der Recurrent Neural Network (RNN)-Schicht zu einer zusammenhängenden Sequenz. [52, Seite 2] Abbildung 10 zeigt beispielhaft wie ein Bild als Eingabe in ein CRNN verwendet wird.

2.3.2. Adaptive Erkennung

Bei der adaptiven Erkennung wird die Erkennung in zwei Durchgängen durchgeführt. In einem ersten Durchgang wird versucht, jedes Wort der Reihe nach zu erkennen. Jedes Wort, das passend zu sein scheint, wird in Form von Trainingsdaten an einen so genannten adaptiven Klassifikator übergeben. Auf diese Weise lernt der adaptive Klassifikator dazu und ist dann in der Lage, die folgenden Texte genauer zu erkennen. Davon profitieren jedoch nur die nachfolgenden Texte. Daher wird ein zweiter Durchlauf der Wörter durchgeführt, um die zunächst schlecht erkannten Wörter besser zu erkennen. [54]

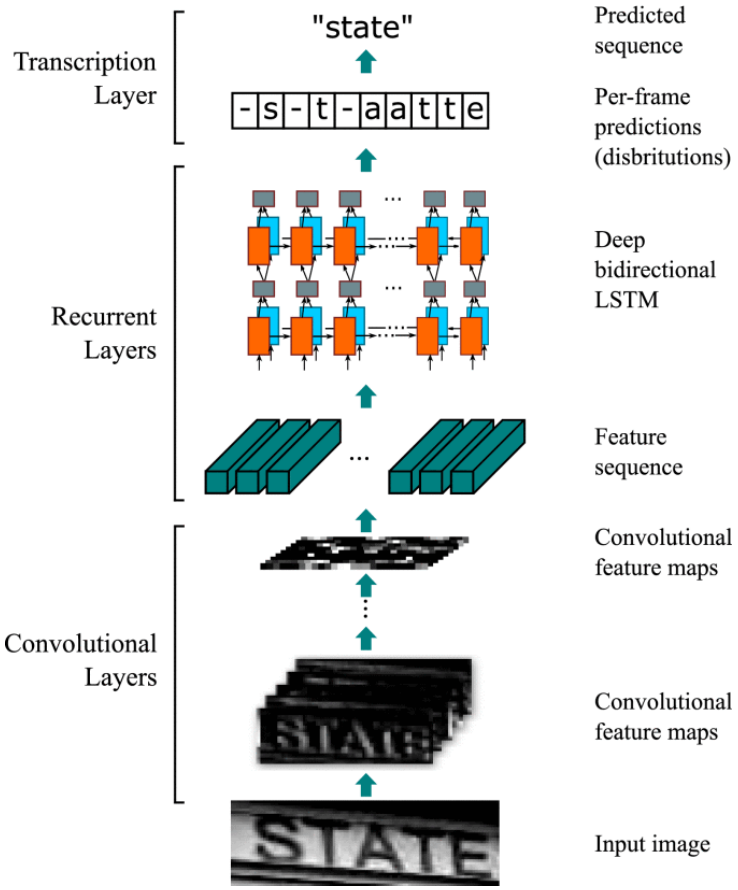


Abbildung 10.: Architektur eines CRNN. [52, Seite 2]

2.3.3. Character Region Awareness For Text Detection

Character Region Awareness For Text Detection (CRAFT) wurde von Baek et al. [8] entwickelt und ist ein Framework zur präzisen Lokalisierung einzelner Zeichen in natürlichen Bildern. Hierfür wurde ein CNN, speziell ein FCN, entworfen, um Zeichenregionen und die Affinität zwischen diesen zu bestimmen.

Die Architektur des neuronalen Netzwerks besitzt auf seiner finalen Ausgabeebene zwei Kanäle, die für die folgenden Score Maps repräsentativ sind: dem `region score` und dem `affinity score`. Der `region score` gibt die Wahrscheinlichkeit an, mit der ein bestimmtes Pixel das Zentrum des Zeichens ist, und der `affinity score` gibt die Wahrscheinlichkeit an, mit der das Zentrum des Raums zwischen benachbarten Zeichen liegt.

In Abbildung 11 ist die Erstellung der beiden Score Maps dargestellt. Zur Bestimmung der Mittelpunkte der Buchstaben und der Affinitätsboxen wird ein Gaußscher isotropischer 2D-Filter in Kombination mit den erfassten Boxen verwendet. Mit Hilfe der perspektivischen Transformation ist es möglich, einen genauen Mittelpunkt zu bestimmen. In Gleichung 2.11 wird die

mathematische Notation des isotropen 2D-Gaußfilters vorgestellt. [13, Seite 580]

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.11}$$

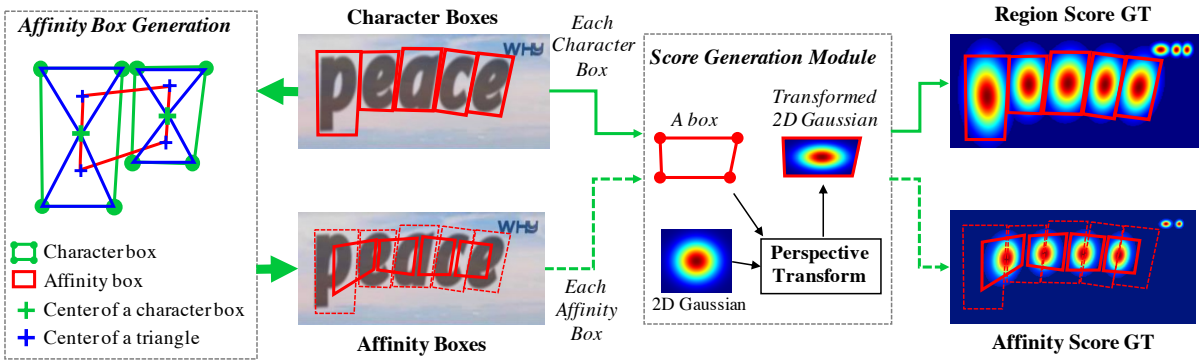


Abbildung 11.: Illustration bezüglich des *affinity* und der *region scores*. (Basierend auf: [8, Seite 3])

Der Vorteil der Texterkennung auf Zeichenebene besteht darin, dass sich die Filter ausschließlich auf die Zeichen und deren Abstände fokussieren können, ohne den gesamten Text zu berücksichtigen.

Die `bounding box` für ein Wort wird einfach durch ein einziges umschließendes Rechteck definiert, das alle `character boxes` einschließt und dabei die `affinity boxes` berücksichtigt.

2.4. Übersetzung von Texten in verschiedene Sprachen

In den vorangegangenen Kapiteln wurde gezeigt, wie mit Hilfe verschiedener neuronaler Netze Text in Bildern erkannt werden kann. Eine weitere Anwendung ist die Übersetzung von Texten in andere Sprachen, die als *Neural Machine Translation (NMT)* bezeichnet wird.

NMT sind mehrschichtige neuronale Netze, die dazu verwendet werden, den Kontext und die Bedeutung von Sätzen zu erfassen und zu übersetzen. Dies wird durch die sogenannte Encoder-Decoder-Architektur erreicht. Das bedeutet, dass eine Sequenz von Eingabewörtern in eine Sequenz von Ausgabewörtern transformiert wird. Aufbauend auf einem RNN gibt es Schichten für den Encoder-Prozess und für den Decoder-Prozess. Der Encoder auf der einen Seite erhält als Eingabe Sequenzen der Eingabesprache und erzeugt als Ausgabe eine kompakte Repräsentation der Eingabesequenz mit dem Ziel, alle Informationen zu einem Ausgabevektor zusammenzufassen. Eine geeignete und konsistente Darstellung des Eingangsvektors für den Decoder zu erhalten ist kompliziert. [9, Seite 2] Der Decoder auf der anderen Seite ist trainiert, das nächste Wort aus dem Eingangsvektor zu extrahieren. Dies geschieht unter Berücksichtigung der bereits ex-

trahierten Wörter und des Eingangsvektors. [9, Seite 3] Wichtig ist auch, dass die Länge der Ein- und Ausgabe festgelegt ist, aber nicht identisch sein muss.

Google hat sein eigenes NMT entwickelt, das als Google Neural Machine Translation (GNMT) bekannt ist. Dieses hat acht Encoder- und acht Decoder-Schichten in seinem Netzwerk. Außerdem verwenden sie den sogenannten Attention-Mechanismus, der die Parallelisierung verbessert und die Trainingszeit verkürzt. Dieser Mechanismus verbindet die unteren Decoder-Schichten mit den oberen Encoder-Schichten. Mit Hilfe dieses Attention-Mechanismus erhält der Decoder die Eigenschaft signifikante Informationen aus dem Enkodierungsprozess, die für den Decodierungsprozess wichtig sein könnten, mit einzubeziehen. Dies erhöht zusätzlich zu den oben genannten Vorteilen die Genauigkeit der Übersetzung. [61]

Es gibt neben Google auch andere Unternehmen, die Übersetzungen als Dienstleistung anbieten. Unter anderem Argos Open Tech mit seinem Open Neural Machine Translation (ONMT). Im Gegensatz zu Googles Lösung handelt es sich dabei um ein Open-Source-System für neuronale maschinelle Übersetzung, das eine vollständige Bibliothek für das Training und die Anwendung neuronaler maschineller Übersetzungsmodelle bietet. Das System wurde vollständig neu geschrieben, um die Effizienz, Lesbarkeit und Verallgemeinerbarkeit zu verbessern. Es enthält sowohl herkömmliche Modelle der neuronalen maschinellen Übersetzung als auch Unterstützung für eine Vielzahl von Optionen, die für eine optimale Leistung erforderlich sind. [25] Mit Hilfe dieser Übersetzungsmodelle kann eine Offline-Übersetzung bereitgestellt werden. Dies öffnet eine Pivot-Übersetzung, die sich automatisch durch die dazwischenliegenden Sprachen bewegt, um zwischen Sprachen zu übersetzen, für die keine direkte Übersetzung existiert. Wenn beispielsweise eine Übersetzung Spanisch-Englisch und Englisch-Französisch vorhanden ist, ist auch eine Übersetzung Spanisch-Französisch möglich. Auf diese Weise ist eine Übersetzung zwischen vielen Sprachen möglich, auch wenn dabei Informationen verloren gehen können. [58]

2.5. Slidebasierte Präsentationsprogramme

Microsofts PowerPoint-Architektur basiert auf dem Standard ECMA-376, der Office Open XML-Vokabulare sowie Darstellungs- und Komprimierungsarten von Dokumenten definiert. [29, 19]

Dies ist hilfreich, um PowerPoint-Präsentationen außerhalb der dafür vorgesehenen Software zu manipulieren. So ist es möglich, eine PowerPoint-Präsentation als Archiv zu öffnen, um auf die darin enthaltenen Informationen zuzugreifen. [31]

Abbildung 12 stellt beispielhaft den Inhalt eines Archivs dar, dies war eine PowerPoint-Präsentation bei der die Dateiendung auf .zip geändert wurde.

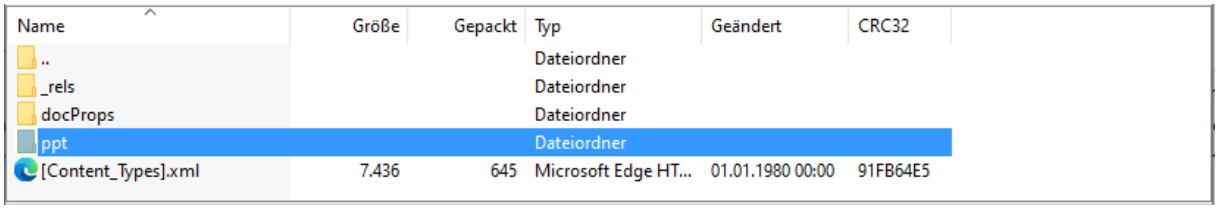


Abbildung 12.: Inhalt einer PowerPoint bei der die Dateiendung auf .zip geändert wurde.

3. Entwicklungansatz des PowerPointDoktors

Dieses Kapitel beschreibt einen Programmieransatz zur Entwicklung des PowerPointDoktors. Die Hauptaufgabe des PowerPointDoktors soll beim Übersetzen von Texten liegen. Standardsoftware ermöglicht es in Teilen diese Funktion abzudecken, jedoch gibt es keine Lösung, welche die funktionale Anforderung Bilder in diesem Übersetzungsprozess einzubeziehen, betrachtet. Somit ist die Entwicklung einer individuellen Softwarelösung erforderlich. Diese soll in diesem Zuge als eine eigenständige Anwendung programmiert werden.

Neben der Übersetzungsgrundfunktion ist es angedacht, dass die Benutzung intuitiv und mit wenigen Klicks bedienbar ist. Dies soll mittels einer übersichtlichen Benutzeroberfläche erzielt werden. Nachfolgend werden noch weitere funktionale Anforderungen definiert und diese genauer beschrieben.

3.1. Übersetzung von PowerPoint-Folien

In PowerPoint-Präsentationen können Texte an verschiedenen Stellen erscheinen. Die tatsächlichen Grenzen liegen bei den Anwendern, die ihre Präsentationen nach Belieben gestalten und mit Inhalten füllen können. Im Folgenden wird Abbildung 13 betrachtet, die den exemplarischen Aufbau einer Folie zeigt, die mit den folgenden Inhalten gefüllt wurde:

- Einem Titel der Folie.
- Einer Aufzählung mit verschiedenen Aufzählungssysmbolen, sowie Textstilen.
- Einem generischen Bild, welches einen Text enthält.
- Zwei Hyperlinks, wobei der Obere einen veränderten Anzeigetext aufweist.
- Notizen, die zusätzliche Informationen und Anmerkungen enthalten.

Neben diesen offensichtlichen Textbereichen gibt es noch weitere Textbereiche, wie zum Beispiel die Fußzeile, die Kopfzeile oder auch die Textbereiche in der Masterfolie. Darüber hinaus können Texte auch in Tabellen, in Diagrammen oder in SmartArt-Objekten vorkommen. Dies sind alles mögliche zu übersetzende Bereiche, die von der Anwendung transformiert werden müssen.

Bilder in Präsentationsfolien sind ein weiterer wichtiger Punkt, der berücksichtigt werden muss. Die Anwendung hat das Ziel, Bilder zu identifizieren. Dazu muss sie auch Textbereiche lokalisieren und extrahieren, was ein wesentlicher Schritt für die spätere Transformation des Bildes ist. Diese extrahierten Texte können auch zur Erkennung von Farben verwendet werden, was es ermöglicht, bei der Erstellung eines übersetzten Bildes einen natürlicheren Einblendungs-Effekt zu erzielen. Dafür werden zwei Farben aus dem darunter liegenden Bereich extrahiert - eine für die Hintergrundfarbe und eine für die Farbe des Textes. Auf der Grundlage dieser Informatio-

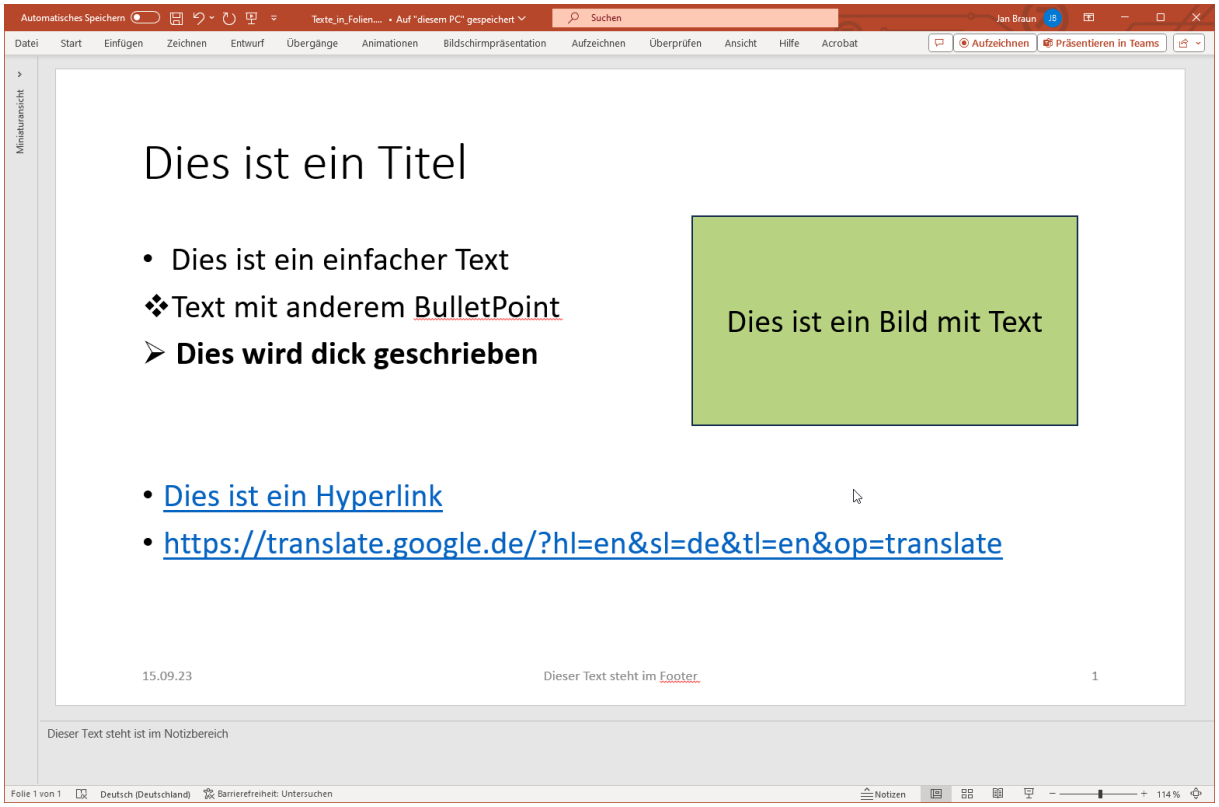


Abbildung 13.: Bildschirmaufnahme einer PowerPoint-Präsentationsfolie.

nen werden neue Teilbilder erstellt, die nur aus dem übersetzten Text bestehen. Diese Teilbilder können dann über die entsprechenden Bildbereiche gelegt werden.

3.2. Mockup zur Benutzeroberfläche

Eine leicht verständliche Benutzeroberfläche trägt wesentlich dazu bei, dass der Umgang mit der Anwendung für die Benutzer komfortabel und ohne großen Schulungsaufwand erlernbar ist. Ihr Ziel ist es, dass der Nutzer sich auf die wesentlichen Aspekte der Anwendung konzentrieren kann. Um eine schnelle Orientierung zu ermöglichen, weist die Benutzeroberfläche eine einheitliche visuelle Struktur mit klaren Informationen auf. Abbildung 14 zeigt ein Mockup der Benutzeroberfläche. Die Anwendung soll drei wesentliche Bereiche bekommen, welche mittels Tabs gegliedert werden. Das macht einen schnellen Wechsel zwischen diesen möglich. Beim Start der Anwendung gelangt der Nutzer in den PowerPoint Scanner Bereich, diese Ansicht besteht aus zwei wesentlichen Bereichen. Der untere Bereich ist anfangs leer und wird erst durch die Interaktion mit dem Ordner-Button mit Leben gefüllt. Der Benutzer wählt zunächst über den Ordner-Button einen Ordner auf seinem System aus. Sobald dies geschehen ist, werden im unteren Bereich PowerPoint-Präsentationen in einer Liste angezeigt. Diese PowerPoint-Präsentationen sind alle

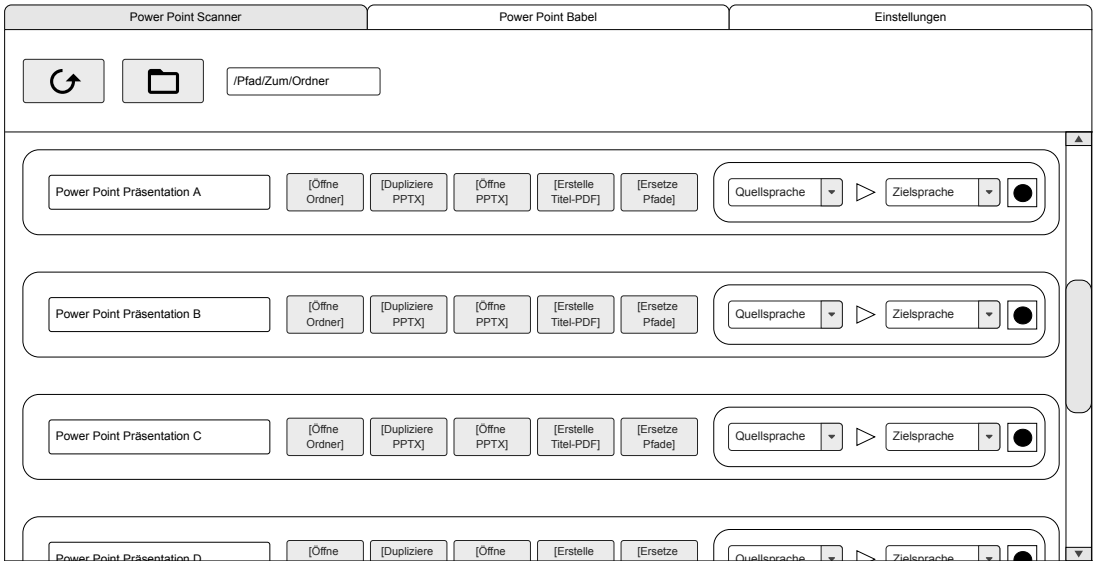


Abbildung 14.: Mockup des ersten Tabs - Zeigt die Anwendung mit selektiertem Ordner. (In Empfehlung von Lehmann)

Präsentationen, die die Anwendung im ausgewählten Ordner und seinen Unterordnern gefunden hat. In der Liste sind neben dem Namen der Präsentation mehrere Buttons enthalten, die es ermöglichen, die Präsentation durch einfaches Anklicken zu bearbeiten beziehungsweise zu modifizieren. Die ersten drei Funktions-Buttons werden folgende Grundfunktionen abdecken:

- Das Öffnen des Ordners in dem die PowerPoint-Präsentation abgelegt ist.
- Das Duplizierung von PowerPoint-Präsentationen.
- Das Öffnen der PowerPoint-Präsentation mit Microsoft PowerPoint.

3.2.1. Extrahieren ausgewählter Folien-Titel

Ziel der vierten Funktion ist das Extrahieren wichtiger Überschriften aus bestimmten Folien durch den Benutzer. Der Benutzer hat somit die Möglichkeit, sein eigenes Inhaltsverzeichnis der PowerPoint-Präsentation als PDF-Dokument zurückzubekommen. Dies erfordert etwas Vorarbeit seitens des Benutzers. Langfristig kann dies jedoch einen Mehrwert für den Benutzer darstellen. Die Idee ist, dass der Benutzer die zusätzliche Arbeit in Form einer selbstständigen Markierung bestimmter Folien ausgeführt hat. Dies kann zum Zeitpunkt der Erstellung oder zum Zeitpunkt der Nachbearbeitung der Folien der Fall sein. Angenommen, die Präsentation besteht aus 100 Folien und der Benutzer benötigt für eine Veranstaltung die Titel der ersten 20 Folien. Um diese relevanten Folien für die Funktion erkennbar zu machen, müsste der Benutzer ein vordefiniertes Symbol zu den Folien hinzufügen und sie damit markieren. Die andere Möglichkeit besteht darin, den entsprechenden Titel in eine Zeichenkette einzukapseln, die vom

Benutzer gewählt werden kann. Denkbar ist auch eine Kombination der beiden Markierungsmöglichkeiten.

3.2.2. Relative Pfade für Video-Dateien setzen

Die fünfte Funktion deckt vor allem bei Präsentationen, die Videos enthalten, einen wichtigen Aspekt der Weitergabe von Präsentationen ab. So ist es für Anwender von Microsoft PowerPoint möglich, Videos direkt an die Präsentation anzuhängen, was zur Folge hätte, dass die Präsentation an Größe zunimmt. Die andere Möglichkeit besteht darin, die Datei zu verlinken.

Microsoft selbst rät dazu, die entsprechenden Dateien für diesen Zweck in den gleichen Ordner zu legen, in dem sich auch die Präsentation befindet, da diese Links kaputt gehen könnten. Jedoch ist die Ablage in den gleichen Ordner nicht zwingend erforderlich. [32]

Ein potenzieller Nutzer könnte die Dateien also auch in einem Unterordner ablegen, was auf dem lokalen Dateisystem des Nutzers unter Umständen kein Problem darstellen würde. Sollte jedoch der Fall eintreten, dass der Benutzer die PowerPoint-Präsentation weitergeben möchte und dabei auf die Unterordner mit den besagten Videodateien zugreift, könnte es dazu kommen, dass die Dateipfade auf dem Zielsystem zu den Videodateien unterbrochen werden. Ziel der Transformierungsfunktion ist daher, die Pfade zu den Videodateien von absoluten in relative Pfade umzuwandeln, was eine fehlerfreie Weitergabe ermöglicht.

3.3. Sprachenverwaltung mit Hinblick auf Offline-Nutzung

Bei der Übersetzung soll auf Online-Übersetzungswerkzeuge zurückgegriffen werden, da die Anbieter in der Regel über sehr gut trainierte Netze verfügen. Dennoch muss sichergestellt werden, dass eine Übersetzung jederzeit möglich ist. Daher wird eine Sammlung von Offline-Sprachmodellen zur Verfügung gestellt, mit deren Hilfe Übersetzungen durchgeführt werden können, ohne dass die Nutzung von Online-Anbietern erforderlich ist.

Dazu wird der zweite Reiter der Anwendung verwendet, der im Mockup in der Abbildung 15 dargestellt ist. Hier steht ganz oben die Möglichkeit zur Auswahl eines Bilderkennungsmodells zur Verfügung, das für die Übersetzung der in Abschnitt 3.1 erwähnten Bilder verwendet wird. In der darunter liegenden Liste sind alle möglichen Sprachmodelle aufgelistet, die mit einem Icon versehen sind, um den Status der jeweiligen Sprache anzuzeigen. Sprachen, die bereits heruntergeladen wurden, sind durch ein Häkchen gekennzeichnet. Ein nach unten zeigender Pfeil markiert verfügbare Sprachen, die noch heruntergeladen werden müssen. Der PowerPointDoktor sollte immer zuerst versuchen, online zu übersetzen und nur im Fehlerfall offline auf die Offline-Sprachmodelle zurückgreifen.

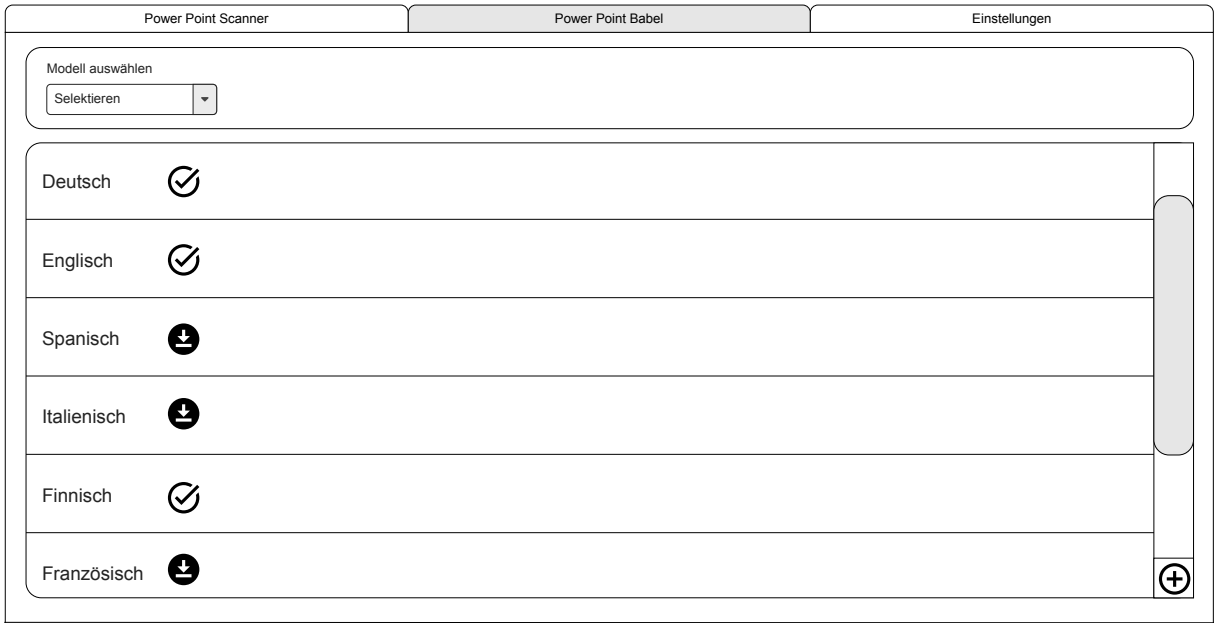


Abbildung 15.: Mockup des zweiten Tabs - Hier können die Sprachen verwaltet werden. (In Empfehlung von Lehmann)

Abbildung 16.: Mockup des dritten Tabs - hier können Einstellungen getroffen werden. (In Empfehlung von Lehmann)

3.4. Einstellungen zur Bedienerleichterung

Das letzte in Abbildung 16 dargestellte Mockup ist der Einstellbereich. Zum Zeitpunkt der Konzeption sind hier nur Parameter für die in Unterabschnitt 3.2.1 genannten Funktionen zum Extrahieren wichtiger Überschriften aus bestimmten Folien vorgesehen. Es ist jedoch durchaus denkbar, dass im Laufe der Implementierung weitere Parameter hinzugefügt werden, um die Nutzung der Anwendung für den Benutzer zu erleichtern.

4. Implementierung des Ansatzes

Wie bereits im einleitenden Kapitel erwähnt, soll eine Desktop-Anwendung erstellt werden, wobei es wichtig ist, dass diese auf einer möglichst großen Anzahl von Endsystemen lauffähig ist. Als Entwicklungssprache wurde Python (Version 3.9) gewählt. Diese Programmiersprache hat den Vorteil, dass die Syntax das Schreiben von sauberem und gut lesbarem Code erleichtert. Dadurch werden die Entwicklungszyklen verkürzt und die Produktivität erhöht. Als Entwicklungseditor wurde Visual Studio Code gewählt.

Python ist eine plattformübergreifende Sprache, was bedeutet, dass Anwendungen, die in Python geschrieben wurden, auf verschiedenen Betriebssystemen wie Windows, Mac OS und Linux ausgeführt werden können. Dies macht sie zu einer geeigneten Sprache für die Entwicklung von Anwendungen, die auf mehr als einer Plattform eingesetzt werden sollen. Um die Entwicklung von Desktop-Anwendungen zu erleichtern, bietet Python verschiedene Graphical User Interface (GUI)-Bibliotheken und Frameworks. [39]

In diesem Kapitel wird die Implementierung des PowerPointDoktors und die damit verbundene Vorgehensweise beschrieben. Zuerst wird die Implementierung des Mockups vorgestellt und erläutert, gefolgt von den grundlegenden Funktionen. Daraufhin wird die Übersetzung von Texten in Bildern behandelt. Darauf aufbauend folgt eine Beschreibung der Erstellung der transformierten PowerPoint-Präsentationen. Abschließend wird die Erstellung der .exe-Datei erläutert.

4.1. Umsetzung der Benutzeroberfläche

Mit Hilfe von Bibliotheken wie Tkinter, PyQt und PySide lassen sich auf unkomplizierte Weise grafische Benutzeroberflächen erstellen und interaktive und vor allem plattformübergreifende Desktop-Anwendungen entwickeln. [39]

Dazu wurde zunächst die Benutzeroberfläche mit PyQt und dem Werkzeug Qt Designer implementiert. Beim Qt Designer handelt es sich um ein Qt-Werkzeug, das für den Entwurf und die Erstellung von grafischen Benutzeroberflächen verwendet wird. Innerhalb dieses Tools werden die Fenster und Dialoge nach dem *What You See Is What You Get (WYSIWYG)*-Prinzip zusammengestellt. Darüber hinaus können diese mit wenigen Änderungen angepasst und getestet werden, um zu sehen, wie sie in verschiedenen Stilen und Auflösungen aussehen werden. [42]

Änderungen an der Umsetzung der Benutzeroberfläche mussten jedoch immer über den Qt Designer erfolgen und anschließend exportiert werden, um sie dann in den Editor zu integrieren. Zudem zeigte sich sehr schnell, dass der automatisch erzeugte Code des Qt Designers unübersichtlich und übermäßig umfangreich war. Aus diesem Grund wurde von PyQt auf PySide6 gewechselt. Dies bedeutet, dass die Definitionen der Benutzeroberfläche mit Hilfe von *Qt Mo-*

deling Language (QML) umgesetzt werden. Bei QML handelt es sich um eine Sprache zur Programmierung von Benutzeroberflächen. Sie ermöglicht es sowohl den Entwicklern als auch den Designern, visuell ansprechende Anwendungen zu erstellen. QML bietet eine gut lesbare, deklarative Syntax, mit der die komplette Benutzeroberfläche in einem *JavaScript Object Notation (JSON)* ähnlichen Format beschrieben wird und die die Verwendung von JavaScript-Ausdrücken unterstützt. [41] Die einzelnen Buttons, Textfelder und andere Oberflächenelemente werden in `QtQuickControls` definiert.

Die Benutzeroberfläche sollte für eine möglichst große Anzahl von Benutzern zugänglich und verständlich sein. Aus diesem Grund wurde ein assoziatives Array im .json Format in die Projektstruktur mit einbezogen. In diesem Array befinden sich Objekte, die nach dem Standard RFC5646 benannt sind. [28] Dieser weitverbreitete Standard wird von vielen Betriebssystemen zur Kennzeichnung von Sprachen verwendet. Bei der Verwendung des Tools wird die Systemsprache des Nutzers ausgelesen und mit den vorhandenen Objektnamen abgeglichen, sodass die entsprechende Sprache ausgewählt wird. Wenn keine der Sprachen passt, wird auf Englisch zurückgegriffen. Zur Zeit werden vom PowerPointDoktor die Sprachen Deutsch, Englisch, Französisch und Spanisch unterstützt. Da die Programmierung auf einem deutschen System erfolgte, zeigen die Abbildungen der Benutzeroberfläche diese Sprache.

Für die Gestaltung der Benutzeroberfläche stehen vordefinierte Stile zur Verfügung. In `QtQuickControls` ist eine große Anzahl von Stilen enthalten, die für jedes System mit Default-Werten belegt sind. Es besteht jedoch auch die Möglichkeit selbst einen Stil auszuwählen. Die Wahl fiel hier auf den Material Style von `QtQuickControls`, welcher den Google Material Design Guidelines folgt. Der Stil ist plattformübergreifend und sieht überall annähernd gleich aus. Eine Abweichung kann durch die verfügbaren Systemschriften und der Art und Weise wie das System diese darstellt entstehen. [40]

Die Farbauswahl erfolgte nach dem in Abbildung 17 dargestellten Muster. Diese Farben wurden während des gesamten Designprozesses in konsistenter Weise verwendet.

Abbildung 17.: Palette der Farben, die in der Benutzeroberfläche verwendet werden.

Die Integration der Benutzeroberfläche erfolgt in der Datei `main.py` der Anwendung. In dieser Datei wird ein neues Objekt einer `QmlApplicationEngine` erstellt, in das dann die Datei `userinterface.qml` geladen wird. Die Benutzeroberfläche ist in Abbildung 18 dargestellt. Sie zeigt den ersten Reiter der Anwendung, welcher entsprechend dem in Abschnitt 3.2 beschriebenen Mockup implementiert wurde.

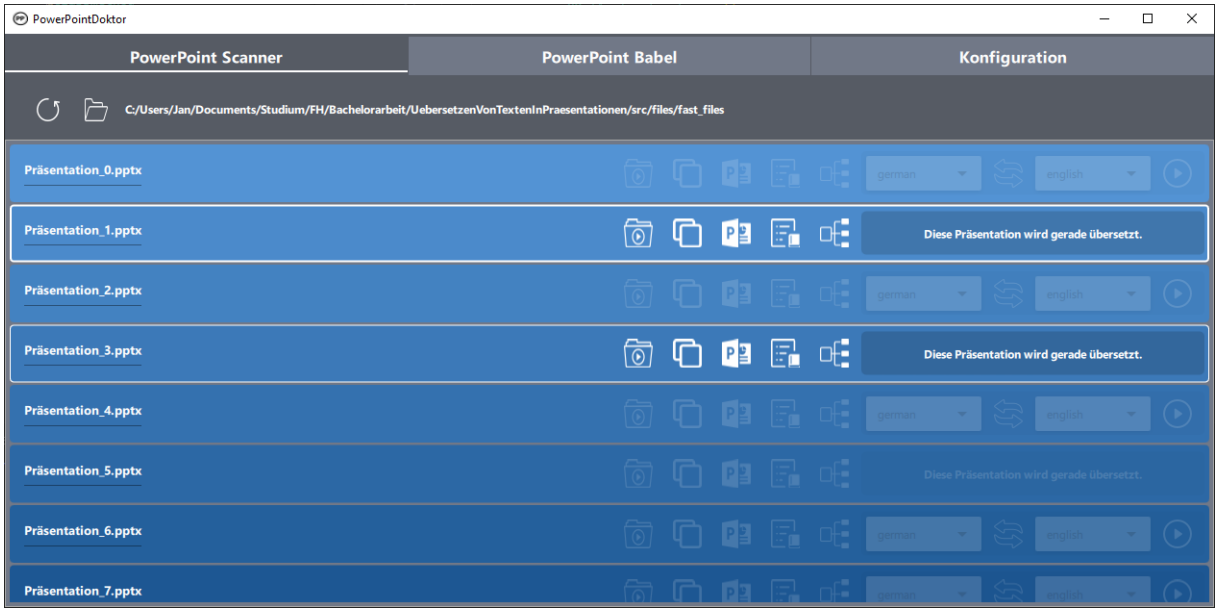


Abbildung 18.: Die Hauptansicht, beziehungsweise erste Reiter des PowerPointDoktors.

Durch eine genaue Aufteilung der beiden in Abbildung 17 gezeigten Blautöne wurde hier der Fading-Effekt der Listenansicht in Abbildung 18 erzielt. Dies erfolgte durch Interpolation. Die Formel für den Farbverlauf lautet $c = a + (b - a) * t$, wobei a für die Startfarbe und b für die Ziel- beziehungsweise Endfarbe steht. Die Variable t enthält einen Wert zwischen 0 und 1, der der Anzahl der zu erzeugenden Farbübergänge entspricht. [65]

Die Liste, die in Abbildung 18 angezeigt wird, basiert auf dem `QAbstractListModel`. Dieses Modell fasst verschiedene Informationen zusammen, die benötigt werden, um die auf dem Bildschirm angezeigte Liste mit Inhalten zu füllen. Beim Design der Liste wurde auf Benutzerfreundlichkeit geachtet. Ein aktives Element in der Liste wird durch den dicken weißen Rahmen dargestellt, in diesem Fall das zweite Element in der gezeigten Grafik. Daneben gibt es noch das aktuell betrachtete Element, über dem die Maus schwebt, das mit einem dünneren Rahmen hervorgehoben wird. Von diesen Element sind die Funktionsschaltflächen deutlich sichtbar. Bei allen anderen Elementen der Liste werden die Funktionsbuttons mit einer geringeren Opazität versehen, um den Fokus auf die genannten Elemente zu unterstreichen. Über das `onEntered()`-Signal einer `MouseArea` erfolgt der Wechsel des aktuell betrachteten Elements. Hierfür besitzt jedes Listenelement eine eigene `MouseArea`. Um dem Benutzer zu beschreiben welche Funktion sich hinter den Funktionsbuttons verbirgt, wird ein ToolTip angezeigt, wenn die Maus über dem Button verweilt. Der String dieser ToolTips ist ebenfalls in dem assoziativen Spracharray enthalten.

Eine neue Funktion, die im Zuge der Implementierung des PowerPointDoktors umgesetzt wurde, ist die Möglichkeit des Umbenennens von PowerPoint-Präsentationen direkt im PowerPointDoktor. Zu diesem Zweck wurde das `QtQuickControl Label` durch ein `TextField` ersetzt.

Um die Umbenennung anzustoßen, verwendet dieses TextField das `onAccepted()`-Signal innerhalb der qml-Datei. Sobald die Benennung abgeschlossen ist, wird ein `onFilenameChanged()`-Signal des Modells ausgegeben. Dies bewirkt ein Update des Modells und damit der Liste.

Neben dieser Funktion führen auch die in Abschnitt 3.2 erwähnte Funktion zum Duplizieren, die Funktion zum Ersetzen von Pfaden aus Unterabschnitt 3.2.2 und die Funktion zum Übersetzen von PowerPoint-Präsentationen zu einer Aktualisierung des Modells.

Zusätzlich wird dem Benutzer des PowerPointDoktors durch Ausblenden der Übersetzungs-Schaltfläche und Einblenden eines Hinweistextes signalisiert, dass die Präsentation gerade übersetzt wird; nach erfolgter Übersetzung wird die Schaltfläche wieder eingeblendet. Davon unberührt bleiben die anderen Funktionsschaltflächen.

Um die Reaktionsfähigkeit der Benutzeroberfläche zu gewährleisten, wurde die Python `Threading` Bibliothek integriert. Mit Hilfe dieser Bibliothek wurden alle Funktionen mit einem gewissen Bedarf an Verarbeitungszeit in einen eigenen `Thread` ausgelagert. Dies umfasst unter anderem die Übersetzung der PowerPoint-Präsentationen, aber auch den Download der Offline-Sprachmodelle, die in dem PowerPoint Babel Bereich, gezeigt in Abbildung 19, in einer Listenansicht dargestellt werden.

Für die Offline-Sprachenliste ist ebenfalls ein `QAbstractListModel` implementiert. Innerhalb dieses Modells wird zum einen die Darstellung des rechts abgebildeten Icons gesteuert und zum anderen der Text für die eigentliche Sprachdarstellung. Welche Daten aus dem Modell an welcher Stelle benötigt werden, wird in der Benutzerschnittstelle mit Hilfe von `ItemDataRoles` umgesetzt. [43] In der Implementierung wurde die `DisplayRole` für die Texte und die `DecorationRole` für die rechtsbündigen Icons verwendet. Bereits heruntergeladene Sprachen sind hier durch ein Häkchensymbol gekennzeichnet, noch herunterladbare Sprachen durch ein nach unten weisendes Pfeil-Symbol und Sprachen, die gerade heruntergeladen werden, durch ein Sanduhr-Symbol.

Der dritte Bereich des PowerPointDoktors wird zur Konfiguration verwendet. Hier wurde die Vorlage aus Abschnitt 3.4 umgesetzt und um einige sinnvolle Konfigurationsmöglichkeiten erweitert. Die Funktion zum Extrahieren ausgewählter Folientitel aus Unterabschnitt 3.2.1 kann sowohl durch die Auswahl einer Form als auch durch die Angabe eines Strings verwendet werden. Des Weiteren ist die Definition einer Standardsprache für die Eingabe und eine Standardsprache für die Ausgabe möglich. Diese wird dann in der ersten Registerkarte auf die einzelnen Elemente der Liste übertragen. Darüber hinaus ist es möglich, die Auswahl der PowerPoint Elemente zu bestimmen, die übersetzt werden sollen. Da die in Abschnitt 3.3 erwähnte Option zur Auswahl der Bilderkennungsmodelle nicht in den Bereich von PowerPoint Babel passt und ein zentraler Bereich für die Konfiguration geschaffen werden soll, wurde sie vom zweiten Reiter der Anwendung hierher verschoben.

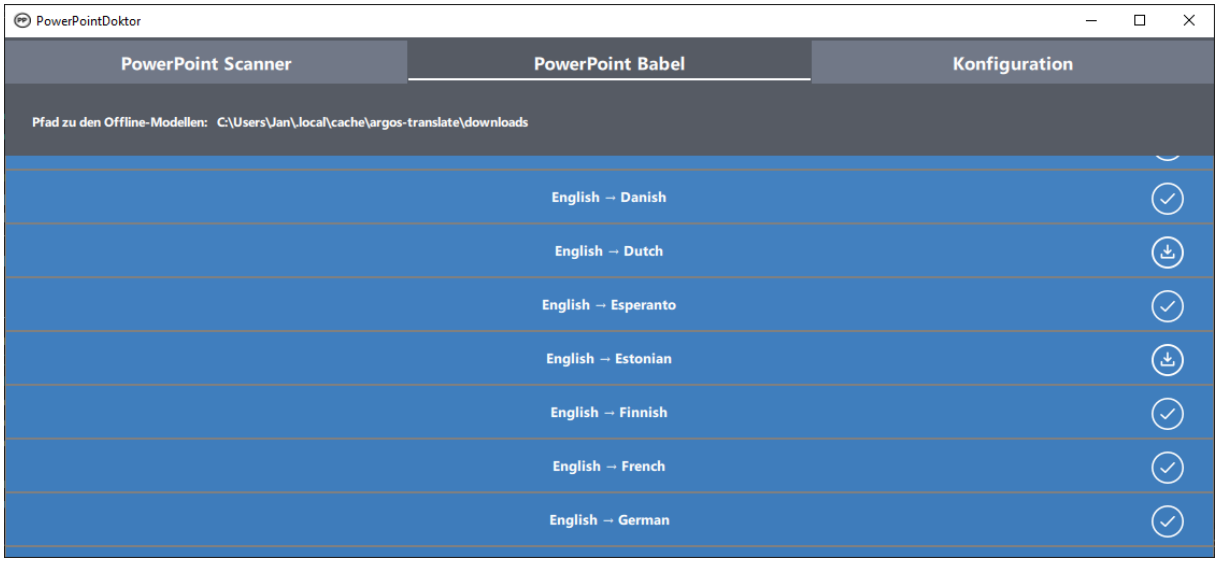


Abbildung 19.: Der zweite Reiter des PowerPointDoktors. Hier können offline Sprachen verwaltet werden.

4.2. Umsetzung der Grundfunktionen

Der PowerPointDoktor ist mit einer Sammlung von Funktionen ausgestattet, die den Umgang mit PowerPoint-Präsentationen erleichtern sollen. Die im vorigen Kapitel erwähnten Funktionen können mit Python-Standardbibliotheken abgedeckt werden. In diesem Kapitel werden nun die Funktionen betrachtet, die sich mit der Modifikation von Präsentationen beschäftigen. Da diese nicht von den Standard Python Bibliotheken abgedeckt werden, wird hierfür die Bibliothek *python-pptx* verwendet, die zum Lesen, Erstellen und Bearbeiten von PowerPoint-Präsentationen verwendet werden kann.

Abbildung 20 zeigt das Klassendiagramm der Implementierung. Im Folgenden wird die `PowerPointHandler`-Klasse betrachtet. In dieser Klasse findet die Modifikation und die Transformation der Präsentationen statt. Zu Beginn liest die Klasse die Präsentation ein und iteriert über die einzelnen Folien.

Das Einlesen der Präsentation als `Presentation`-Objekt der *python-pptx* Bibliothek ermöglicht den Zugriff auf viele weitere Funktionen und Attribute. Darunter auch das `Slides`-Objekt. In diesem Objekt sind alle Folien der eingelesenen Präsentation enthalten, über die zur weiteren Bearbeitung iteriert wird. Des Weiteren verfügt jede so genannte Folie über ein `Shapes`-Objekt. Dieses enthält alle `Shapes` der jeweiligen Folie. [16] Die *python-pptx* Bibliothek unterteilt diese in verschiedene `Shapes`. Relevant sind die Folgenden:

- `auto shape` – Dies ist eine gewöhnliche Form, beispielsweise ein Rechteck, eine Ellipse oder ein Pfeil. Es gibt ca. 180 verschiedene Formen, die auch Text enthalten können. [33]

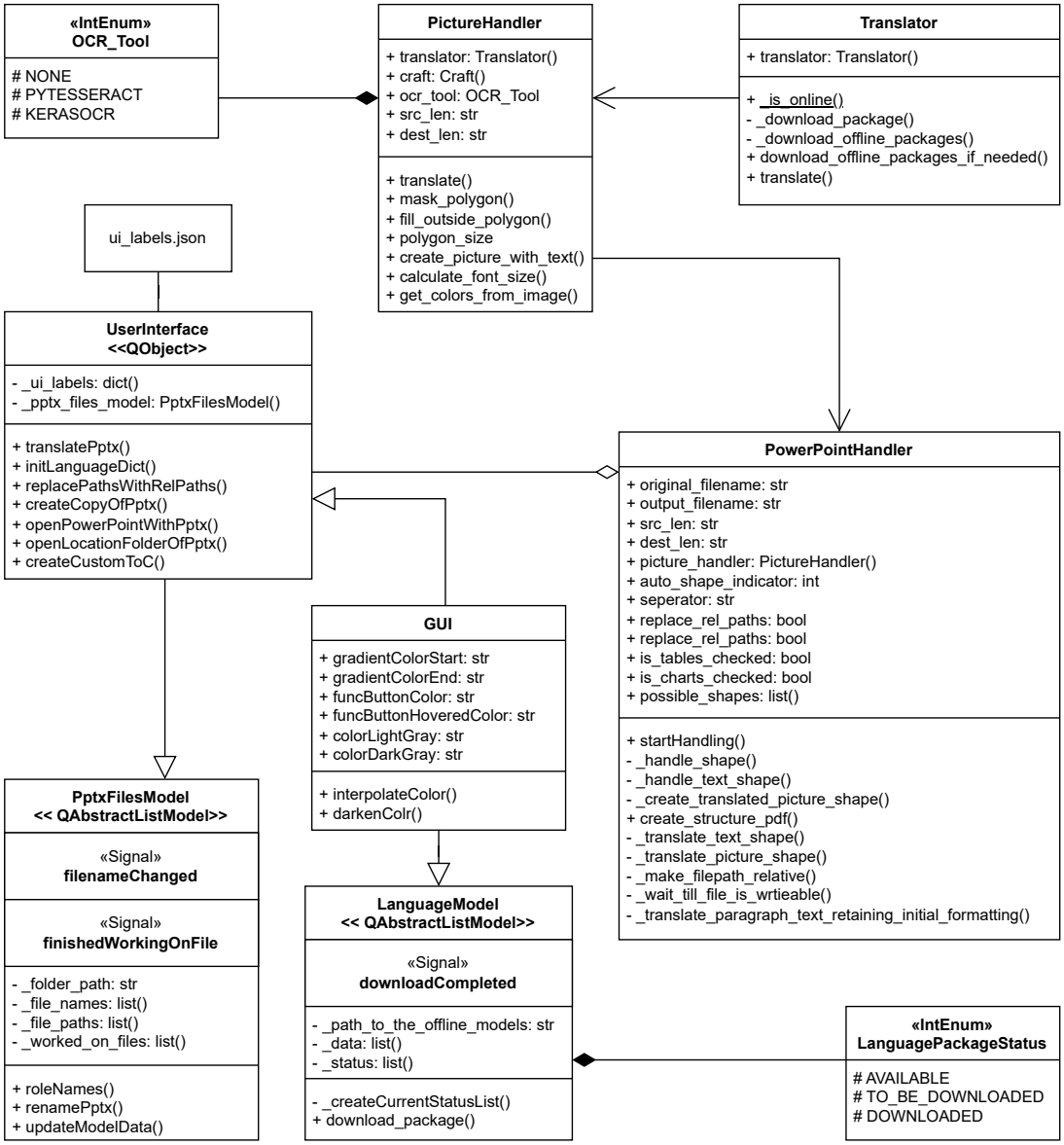


Abbildung 20.: UML-Klassen-Diagramm des PowerPointDoktors.

- `group shape` - In PowerPoint können mehrere Formen gruppiert werden, sodass sie gemeinsam ausgewählt, verschoben, skaliert und sogar befüllt werden können. Das `group shape` wird erstellt, wenn mehrere `Shapes` gruppiert werden. Ein sogenannter Selektierungsrahmen wird außerdem angezeigt, sobald ein `Shape` aus dem `group shape` selektiert wurde.
- `placeholders` – Hierbei handelt es sich auch um `auto shapes`, diese können Teil eines Folienlayouts oder des Masterfolien Layouts sein und von den Folien geerbt werden. Formen übernehmen beim Hinzufügen von Inhalten die Formatierung des `placeholders`.

- `picture` - Ein Rasterbild, zum Beispiel ein Foto oder eine Clip-Art, wird in PowerPoint als Bild bezeichnet. Es handelt sich um eine eigene Art von `Shape`, die sich anders verhält als ein `auto shape`. Eine AutoForm kann auch eine Bild-Füllung enthalten, bei der ein Bild als Hintergrund der Form erscheint.

Auf dieser Basis ist es möglich, eine Reihe realer `Shapes`, die von Microsoft bekannt sind, darzustellen. Dazu gehören `shape shapes`, `text boxes` und `placeholder`. Außerdem unterstützt die Bibliothek das `media clip shape`.

Nachfolgend wird das Ersetzen von absoluten Pfaden durch relative Pfade aus Unterabschnitt 3.2.2 und das Erstellen der extrahierten Titel PDF aus Unterabschnitt 3.2.1 behandelt. Hierfür werden die `Shapes media clip`, `shape shapes` und `text boxes` benötigt.

Dazu wird eine neue lokale PowerPoint-Präsentation erstellt und auf einer beliebigen Seite wird ein Link zu einer lokalen Videodatei eingefügt. Ausgehend von Abschnitt 2.5 ist bekannt, dass es möglich ist, den Inhalt einer Präsentation genauer zu untersuchen, indem sie in ein Archiv umgewandelt wird. Anschließend werden die Pfade der verlinkten Dateien in der entsprechenden Datei betrachtet. Diese werden in Listing 4.1 angezeigt.

```
1 <?xml version="1.0" encoding="utf-8" standalone="true"?>
2 <Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
3     <Relationship
4     Target="file:///C:\Users\Jan\Documents\PowerPointDoktor\src\files\movie.mp4"
5     Type="http://schemas.microsoft.com/office/2007/relationships/media"
6     Id="rId3"
7     TargetMode="External" />
8 </Relationships>
```

Listing 4.1: Inhalt der Datei *presentation.xml.rels*, die geöffnet wurde, nachdem die .pptx-Datei in ein Archiv umgewandelt wurde.

Relevant für die Implementierung sind alle dargestellten Attribute der Relation. Dass es sich bei den Pfaden um absolute Pfade zu den Dateien handelt, ist hier direkt ersichtlich. Weiterhin sehen wir, dass das `media shape` verwendet wurde. Bei der Iteration über die einzelnen `Shapes` auf einer Folie greift die Logik also gezielt nur auf das `media shape`.

Die Abfrage, ob ein externer Link vorliegt oder nicht, wird wie in Listing 4.2 dargestellt. Auf diese Weise wird sichergestellt, dass sich nur die verlinkten Medien verändern. PowerPoint unterscheidet zwischen einem eingebetteten Video und einem Link zu einem Video, das auf dem Computer gespeichert ist. [32]

Für die nachfolgende Ersetzung wird diese Information mit `multimedia_paths_to_be_changed.append([rel.reltype, new_video_path, rel.rId])` für alle in der

```
286     # update relative paths to included file references
287     elif replace_rel_paths is not None and shape.shape_type == MSO_SHAPE_TYPE.MEDIA:
288         # iterate over the relationships
289         for rel in shape.part.rels:
290             # we only want to change the path if it is acutally external,
291             # else the media is already packed within the powerpoint.
292             if (rel.is_external):
293                 new_video_path = self._make_filepath_relative(rel)
294                 multimedia_paths_to_be_changed.append([rel.reltype, new_video_path, rel.rId])
```

Listing 4.2: Programmauszug aus dem `PowerPointHandler`. Hier greift die Logik zum Ersetzen der Pfade.

Präsentation gefundenen rel-Objekte in einer Liste gespeichert. Anschließend werden sie ersetzt und eine neue PowerPoint-Präsentation im selben Ordner erzeugt.

Nun wird die Funktionsimplementierung zum Extrahieren ausgewählter Folientitel aus Unterabschnitt 3.2.1 betrachtet. Der Benutzer gibt einen String und/oder eine `auto shape` über die in Abbildung 21 abgebildete Eingabemaske ein. Während des Iterationsprozesses über die `Shapes` prüft der Algorithmus, ob die eingegebene Zeichenkette Teil eines gefundenen Textes ist. Ist dies der Fall, wird der Titel und die Foliennummer der aktuell betrachteten `slide` einem *Dictionary* hinzugefügt. Analog dazu erfolgt ein Abgleich der `Shapes` mit dem `MSO_SHAPE_TYPE.AUTO_SHAPE`. Wenn dies wahr ist, wird das Attribute `shape.auto_shape_type` mit dem entsprechenden `AUTO_SHAPE` befüllt sein. Es wird dann überprüft, ob es sich um einen der vom PowerPointDoktor erlaubten `AUTO_SHAPE` handelt.

- MSO_AUTO_SHAPE_TYPE.RECTANGLE
- MSO_AUTO_SHAPE_TYPE.CROSS
- MSO_AUTO_SHAPE_TYPE.HEART
- MSO_AUTO_SHAPE_TYPE.DIAMOND

In diesem Fall wird der Titel ebenfalls zum *Dictionary* hinzugefügt. Nach Betrachtung aller Folien wird mit dem *Dictionary* und PyFPDF ein PDF-Dokument mit den gesammelten Titeln erstellt. PyFPDF ist eine Bibliothek zur Erstellung von PDF-Dokumenten unter Python. Das PDF-Dokument wird im gleichen Ordner wie die PowerPoint-Präsentation erstellt und dem Benutzer direkt nach der Erstellung angezeigt. Weiterhin wird ein direkter Text-Link, der das Öffnen der dazugehörigen PowerPoint-Präsentation ermöglicht, in der ersten Zeile nach der Überschrift dargestellt.

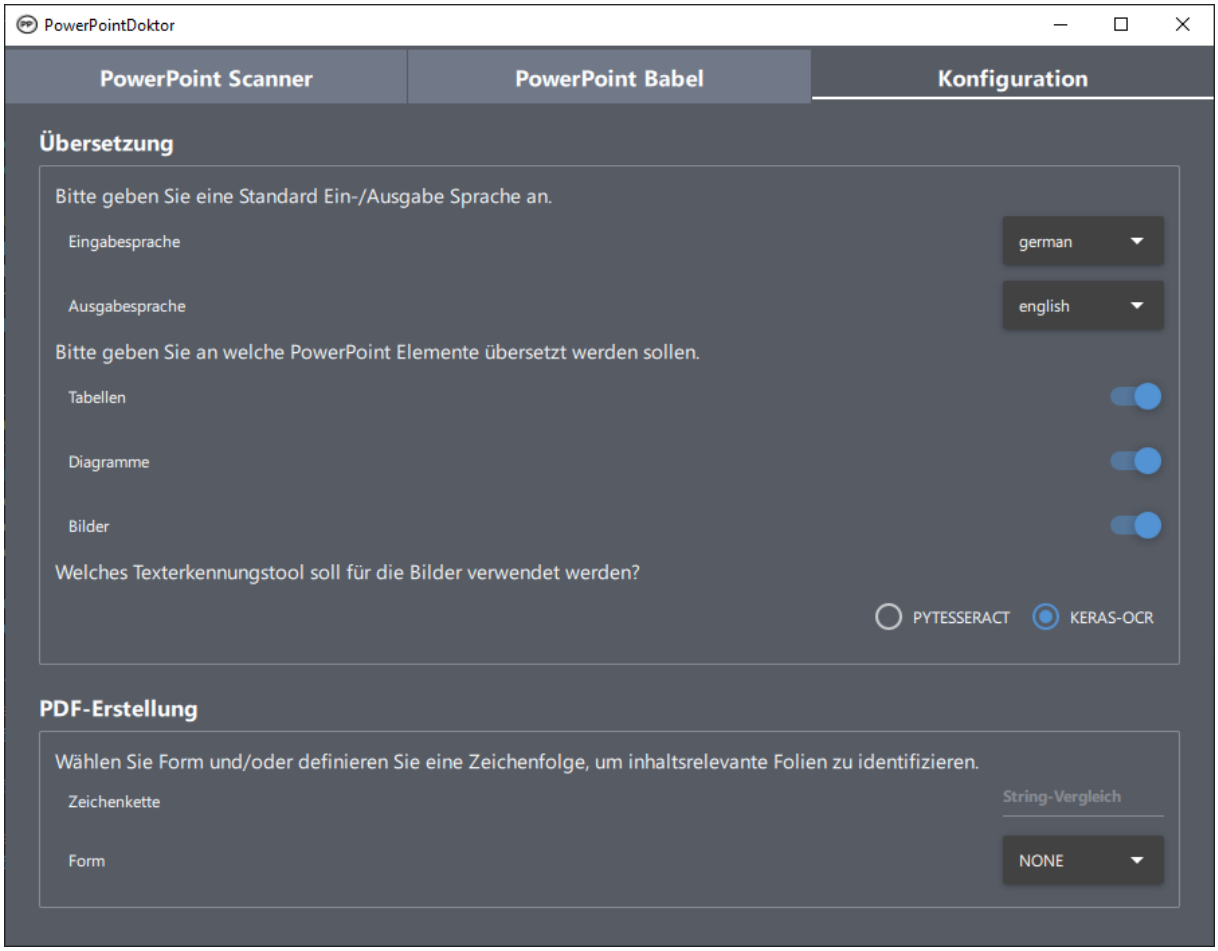


Abbildung 21.: Im dritten Reiter kann der PowerPointDoktor eingestellt werden.

4.2.1. Übersetzen der Textelemente in einer PowerPoint-Folie

Um alle möglichen `Shapes` abzudecken, die Text Shapes enthalten können, wird während der Iteration durch die `Shapes` ein Abgleich mit einer Enumeration der *python-pptx* Bibliothek durchgeführt. In dieser `MSO_SHAPE_TYPE` Enumeration werden die möglichen Microsoft Office `Shape` Typen dargestellt. [15]. Der PowerPointDoktor berücksichtigt die folgende bei der Betrachtung von zu übersetzenden Texten:

- `MSO_SHAPE_TYPE.GROUP` - Hierbei handelt es sich um eine Gruppierung von `Shapes`, die beliebig tief gekapselt werden kann. Die Anwendung geht rekursiv durch jede Ebene und bearbeitet alle *Shapes* in dieser Liste mit möglichen text frames.
- `MSO_SHAPE_TYPE.TABLE` - Eine normale Tabelle mit Zellen. Hier werden die Zellen durchlaufen und die Texte übersetzt.
- `MSO_SHAPE_TYPE.CHART` - Im Gegensatz zur Tabelle ist dieses `Shape` komplizierter zu hand-

haben. Wenn die Tabelle einen Titel hat, dann wird dieser Titel übersetzt. Darüber hinaus kann ein Chart so genannte Plots enthalten, in denen Kategorien enthalten sind. Dies sind die Beschriftungen des Diagramms, zum Beispiel bei einem Kreisdiagramm oder einem Balkendiagramm. Weiterhin kann es neben den Kategorien auch sogenannte Serien geben. Diese befinden sich direkt an der visuellen Darstellung. Hierbei handelt es sich in der Regel um Zahlen und Werte, die keine Übersetzung benötigen.

- `MSO_SHAPE_TYPE.PICTURE` - Ein eingefügtes Bild. Wie dieses `Shape` behandelt wird, ist im Abschnitt 4.3 beschrieben.

Außerdem ist der gefundene Text hierarchisch gegliedert. An oberster Stelle befindet sich das `text_frame`, welches aus `paragraphs` besteht, wiederum bestehen `paragraphs` aus `runs`. Alle diese Objekte besitzen ein `text` Attribut, dieses unterscheidet sich wie folgt:

1. `Shape.text_frame` - Der Text des `text_frames` enthält den gesamten Text aller seiner `paragraphs`, die durch die entsprechenden Zeilenumbrüche zwischen den Absätzen verbunden sind.
2. `TextFrame.paragraphs` - Der Text des `paragraphs` ist die Gesamtheit des Textes aller `runs`, die zu einer langen Zeichenkette verkettet sind. Für jeden sogenannten weichen Zeilenumbruch wird ein vertikales Tabulatorzeichen gesetzt. Ein weicher Zeilenumbruch ist wie ein Zeilenumbruch, aber ohne Absatzende. Jeder `paragraph` besitzt ein `font`-Objekt, das die Standard-Font-Eigenschaften für die `runs` dieses `paragraphs` enthält.
3. `_Paragraph.runs` - Dies ist die unterste Ebene, jeder `paragraph` kann aus mehreren `runs` bestehen, die zusätzlich ein eigenes `font`-Objekt enthalten können, das ein von der Standardkonfiguration abweichendes Styling definiert.

Der PowerPointDoktor übersetzt die Texte auf der `run`-Ebene, gezeigt in Listing 4.3, sodass sichergestellt ist, dass die Konfiguration der Schrift erhalten bleibt.

```
362     def _translate_paragraph_text_retaining_initial_formatting(self, paragraph):
363         # iterate of the paragraph runs
364         for idx, run in enumerate(paragraph.runs):
365             # needs that spacing afterwards or the strings will be tied together
366             paragraph.runs[idx].text = self.picture_translator.translator
367                     .translate(run.text, self.src_len, self.dest_len) + " "
```

Listing 4.3: Programmauszug aus dem `PowerPointHandler`. Hier greift die Logik zum Übersetzen auf `run`-Ebene.

Nachfolgend ein Beispiel, um die Funktionsweise besser verstehen zu können. Der folgende Text:

Einleitung: *Hier ein kurzes Beispiel für einen Text.*

wird vom PowerPointDoktor nach der Übersetzung so angezeigt:

Introduction: *Here is a short example of a text.*

Im Gegensatz dazu würde der Text auf den Folien wie folgt aussehen, wenn die Übersetzung auf der Ebene `paragraphs` erfolgen würde:

Introduction: Here is a short example of a text.

Nachdem die zu übersetzenden Texte definiert wurden, wird nun die `Translator`-Klasse aus dem Klassen-Diagramm in Abbildung 20 betrachtet. In dieser Klasse wird die eigentliche Übersetzung der Strings durchgeführt. Der PowerPointDoktor verwendet hierfür `googletrans`, welcher kompatibel mit den Python Versionen 3.6 und höher ist. Hierbei handelt es sich um eine kostenlose Python-Bibliothek, die die Google Translate Ajax API implementiert, um Methoden aufzurufen, die Textübersetzungs- oder Spracherkennungsfunktionen bereitstellen. Die maximale Zeichenanzahl pro Aufruf beträgt 15.000 Zeichen. Außerdem erfolgt der Aufruf über die gleichen Server wie bei `translate.google.com` und ist daher sehr schnell und zuverlässig. [20] Voraussetzung ist allerdings, dass der PowerPointDoktor auf einem Computer mit Internetverbindung ausgeführt wird.

Um dies sicherzustellen, wird die in Listing 4.4 gezeigte Funktion `_is_online()` verwendet. Hier wird die Onlineverbindung über einen HTTPS-Request an `translate.google.com` gesendet. Ist keine Internetverbindung vorhanden, wird bei diesem Aufruf eine `RequestException` erzeugt.

```
13  @classmethod
14  def _is_online(cls):
15      try:
16          requests.get('https://translate.google.com')
17          return True
18      except requests.exceptions.RequestException:
19          return False
```

Listing 4.4: Programmauszug aus der `Translator`-Klasse. Die Funktion `_is_online()` wird genutzt um zu Überprüfen ob eine Verbindung zum Internet besteht.

Da jeder`_Paragraph.runs.text` einzeln betrachtet wird, muss für jeden dieser Texte ein eigener Übersetzungsaufruf und damit eine Online-Konnektivitätsprüfung erfolgen.

Im Offline-Fall wird, wie in Listing 4.5 gezeigt, auf den Übersetzer der Bibliothek `argostranslate` zurückgegriffen. `Argostranslate` ist eine in der Programmiersprache Python geschriebene Open-Source-Bibliothek für Offline-Übersetzer. Der Hauptvorteil dieser Bibliothek

```
51      def translate(self, text, src_len, dest_len):
52          # check if the text is consisting entirely of blank characters
53          if text.isspace():
54              return text
55          # check if we are online and use the corresponding translator
56          if self._is_online():
57              return self.translator.translate(text, src=src_len, dest=dest_len).text
58          else:
59              return translate.translate(text, src_len, dest_len)
```

Listing 4.5: Programmauszug aus der `Translator`-Klasse. Die Funktion `translate()` zeigt wie der PowerPointDoktor die Unterscheidung zwischen Offline/Online-Übersetzung handhabt.

besteht darin, dass sie die Installation von Sprachmodellpaketen unterstützt, die als Zip-Archive mit der Endung ".argosmodel" vorliegen und die für die Übersetzung benötigten Daten enthalten. [58] Die Verwaltung dieser Offline-Sprachmodelle erfolgt im zweiten Reiter des PowerPointDoktors (Abbildung 19).

4.3. Von identifizierten Bildern zu übersetzten Bildern

In diesem Abschnitt wird der `PictureHandler`-Klasse aus Abbildung 20 betrachtet, diese dient hauptsächlich der Übersetzung der `MSO_SHAPE_TYPE.PICTURE` aus den Folien. Da sich der Prozess nun im Stadium befindet, in dem bekannt ist, dass ein `Picture Shape` gefunden wurde, kann mit dieser Information auf die Eigenschaft `blob` des Bildes zugegriffen werden. Dies ist ein binärer Bytestream des Bildes. [14] Nun muss an dieser Stelle Vorarbeit geleistet werden, um weitere Verarbeitungsschritte zu ermöglichen. Zuerst wird das Array mit `np.frombuffer()` in ein `np.array` und dann mit `cv2.imdecode` in ein Bild umgewandelt. Anschließend werden die Farbkanäle mit `cv2.cvtColor()` getauscht und das Bild mit `Image.fromarray().save()` gespeichert, bevor es an den `PictureHandler` übergeben wird. Der Prozess der Transformation wird im weiteren Verlauf dieses Unterkapitels noch genauer beschrieben und kann zum besseren Verständnis in Form eines Flussdiagramms in Abbildung 22 betrachtet werden.

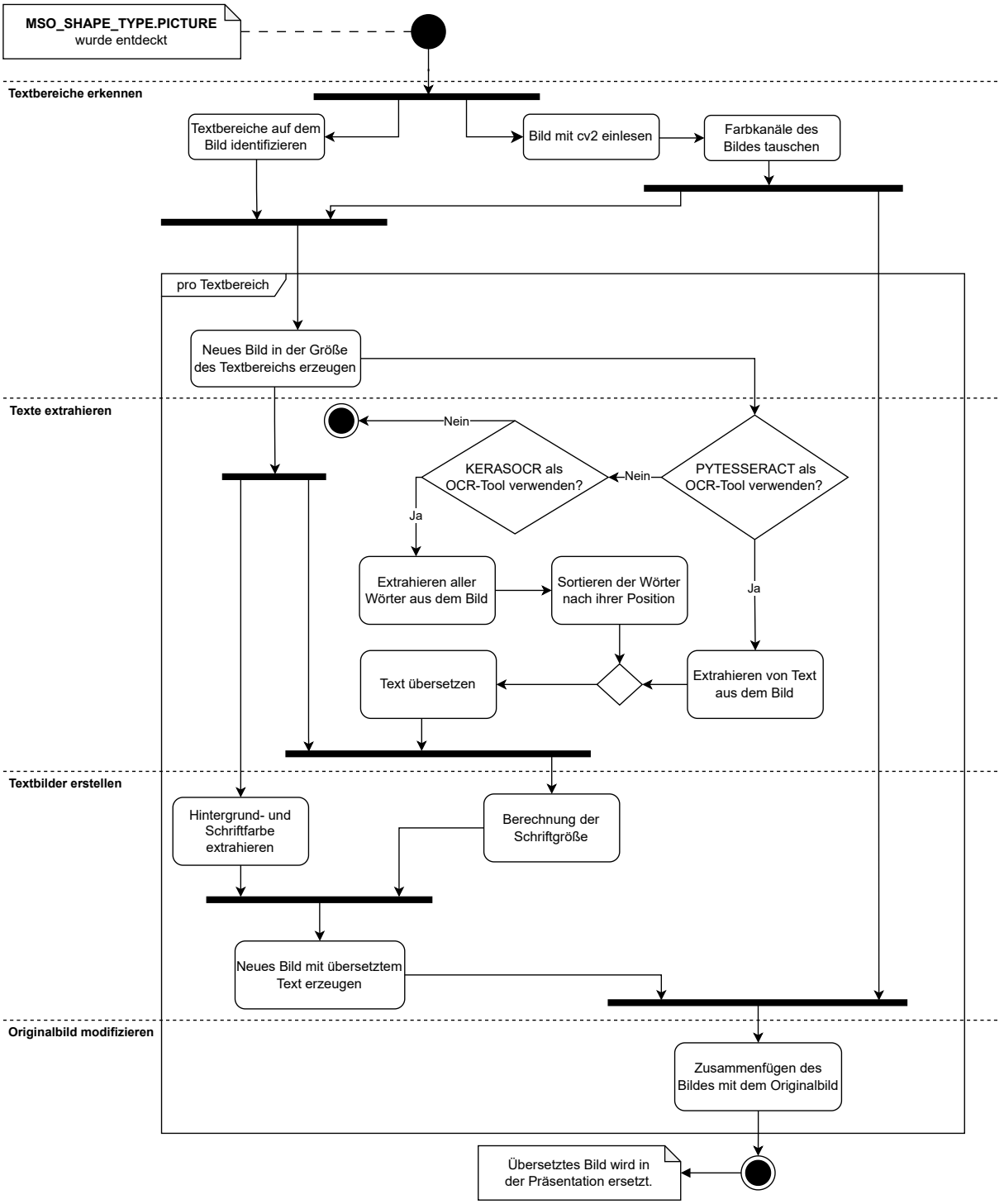


Abbildung 22.: UML-Flussdiagramm für die Übersetzung eines Bildes.

Textbereiche erkennen

Um Textbereiche zu identifizieren, wird das in Unterabschnitt 2.3.3 gezeigte CRAFT-Framework verwendet. Der in Abbildung 20 gezeigte `PictureHandler` verfügt zu diesem Zweck über ein `craft`-Objekt. Mit diesem wird der eigentliche Detektionsalgorithmus aufgerufen, was über `self.craft.detect_text()` geschieht. Diese Funktion liefert ein *Dictionary* mit unterschiedlichen Ausgabewerten zurück. Wichtig ist hier der Eintrag `boxes`. Dieser enthält eine Liste von Koordinaten der Punkte der vorhergesagten `bounding boxes`. Bevor über die einzelnen Textbereiche iteriert werden kann, wird das Bild mit `cv2.imread()` eingelesen. Da das Bild standardmäßig im BGR-Farbraum gespeichert wird, müssen anschließend die Farbkanäle mit `cv2.cvtColor()` von BGR auf RGB umgestellt werden. Nun werden die `bounding boxes` betrachtet. Dafür wird ein neues Bild erstellt, das nur aus den Koordinaten der Punkte aus den `bounding boxes` besteht.

Texte extrahieren

Im Folgenden soll die Extraktion der Texte aus den vorbereiteten Teilbildern betrachtet werden. Je nachdem, welches Texterkennungswerkzeug der Benutzer in den Einstellungen von Abbildung 21 ausgewählt hat, wird entweder `PYTESSERACT` oder `KERAS_OCR` verwendet. `KERAS_OCR` ist ein Framework, das den in Unterabschnitt 2.3.3 vorgestellten Textdetektor CRAFT und eine eigenes CRNN verwendet, um Texte zu extrahieren. `PYTESSERACT` hingegen arbeitet mit der aus Unterabschnitt 2.3.2 vorgestellten adaptiven Erkennung. Im Gegensatz zu `KERAS_OCR` kann `PYTESSERACT` mit `pytesseract.image_to_string()` direkt ganze Sätze zurückgeben, während `KERAS_OCR` mit `pipeline.recognize()` eine Liste von Listen zurückgibt, die aus einzelnen Wörtern und ihren `bounding boxes` beziehungsweise ihrer Position im Bild besteht. Diese Informationen werden verwendet um die Liste zu sortieren und einen String aus den gefundenen Wörtern zu erzeugen. Der letzte Schritt in diesem Abschnitt besteht darin, den extrahierten Text mit Hilfe der in Unterabschnitt 4.2.1 beschriebenen Übersetzungslogik zu übersetzen.

Textbilder erstellen

Um eine natürlichwirkende Ersetzung der Textbereiche im ursprünglichen Bild zu erreichen, müssen neue Bilder, die hier Textbilder genannt werden, erzeugt werden. Dabei werden neben dem übersetzten Text auch Informationen über die Schriftfarbe, die Hintergrundfarbe und die Schriftgröße berücksichtigt.

Der in Listing 4.6 gezeigte Programmauszug zeigt wie die Farben mit dem bekannten Clustering-Algorithmus `KMeans` extrahiert werden. Beim Clustern werden Datenpunkte aufgrund ihrer Ähnlichkeit in eine bestimmte Anzahl von Gruppen zusammengefasst. Dieses Verfahren kann genutzt werden, um Bilder zu komprimieren und auf eine bestimmte Anzahl `k` von Farben zu reduzieren.

```
273     def get_colors_from_image(self, image):
274         clt = KMeans(n_clusters=2)
275         clt.fit(np.reshape(image, (-1, 3)))
276 
277         n_pixels = len(clt.labels_)
278         # count pixels per cluster
279         counter = Counter(clt.labels_)
280         perc = {}
281         for i in counter:
282             perc[i] = np.round(counter[i]/n_pixels, 2)
283         perc = dict(sorted(perc.items()))
284 
285         b, f = clt.cluster_centers_[:2]
286         b1, b2, b3 = map(int, b)
287         f1, f2, f3 = map(int, f)
288 
289         # return dominant color, which should be the background color
290         if perc[0] > perc[1]:
291             return (b1, b2, b3), (f1, f2, f3)
292         else:
293             return (f1, f2, f3), (b1, b2, b3)
```

Listing 4.6: Programmauszug aus der `PictureHandler`-Klasse. Die Funktion `get_colors_from_image()` zeigt wie die Hintergrund- und Schriftfarbe mittels KMeans extrahiert werden.

Um eine Farbe für den Text und den Hintergrund zu erhalten, wird `k=2` gesetzt. Abschließend wird überprüft, wie hoch die Anteile der beiden Farben ist um so sicherzustellen, dass die Rückgabefarbe passend ist. Es wird angenommen, dass die Hintergrundfarbe den größeren Anteil an Farbe einnimmt.

Zur Bestimmung der geeigneten Schriftgröße wird ein direkter Ansatz gewählt. Dazu wird die Höhe und Breite der `bounding box` des Textes benötigt, sowie der Text in einer Standardschriftart, die hier per Default auf Arial gesetzt ist. Der Algorithmus in Listing 4.7 beginnt mit einer Schriftgröße von 1 und prüft mit der Funktion `font.getsize()`, ob die `bounding box` verletzt würde, wenn der Text in Schriftgröße 1 erstellt werden würde. Als `bp_height` und `bp_width` werden hier Höhe und Breite der `bounding box` angegeben. Wenn es keine Verletzung gibt, wird die Schriftgröße um den Wert `steps` erhöht und die Überprüfung erneut ausgeführt. Wird jedoch die `bounding box` verletzt, wird die Variable `steps` halbiert und die ursprüngliche Schriftgröße um den neuen `steps` Wert verkleinert. Auf diese Art und Weise ist es möglich, die geeignete Schriftgröße zu bestimmen.

```
259     # scale text size in relation to the size of the image
260     while True:
261         if font.getsize(text)[1] < bp_height and font.getsize(text)[0] < bp_width:
262             fontsize += steps
263         else:
264             steps = steps // 2
265             fontsize -= steps
266         font = ImageFont.truetype(font_path, fontsize)
267         if steps <= 1:
268             break
```

Listing 4.7: Programmauszug aus der `PictureHandler`-Klasse. Diese Logik wird zur Berechnung der geeigneten Schriftgröße verwendet. (Basierend auf: [36])

Nachdem alle Informationen für die Erzeugung des Bildes vorliegen, wird ein neues Textbild mit `Image.new('RGB', (int(width), int(height)), color = background_color)` erzeugt. Nun fehlt noch der übersetzte Text, welcher nach der Berechnung des Textzentrums mittels `draw.text((x, y), text, font=font, fill=font_color)` in das Bild eingefügt wird. Das `draw` Objekt ist hierbei eine Instanz der Klasse `ImageDraw.Draw(img)`.

Originalbild modifizieren

Am Ende der Iterationsschleife wird das Textbild in das Originalbild eingefügt. Dies geschieht für jede `bounding box`, wie in Listing 4.8 gezeigt, bis alle Textbereiche des Bildes ersetzt wurden. Schließlich wird das Bild zurückgegeben und in die Präsentation in der `PowerPointHandler`-Klasse eingefügt.

```
150     y1, y2 = bounding_box_y, bounding_box_y + text_image.height
151     x1, x2 = bounding_box_x, bounding_box_x + text_image.width
152     original_image[y1: y2, x1: x2] = text_image
```

Listing 4.8: Programmauszug aus der `PictureHandler`-Klasse. Es zeigt, wie der Bereich des Originalbildes durch das neu erstellte Textbild ersetzt wird.

4.4. Auslieferung des PowerPointDoktors

Der PowerPointDoktor soll, wie in Kapitel 3 beschrieben, als eine einzige ausführbare Datei ausgeliefert werden. Für diesen Auslieferungsprozess wurde die Bibliothek *PyInstaller* verwendet, die ein in Python geschriebenes Programm einliest. Sie analysiert den Code, um alle anderen Module und Skripte zu ermitteln, die für die Ausführung der Anwendung erforderlich sind, erstellt Kopien all dieser Dateien und legt sie zusammen mit dem Skript in einem einzigen Verzeichnis oder, falls konfiguriert, in einer einzigen ausführbaren Datei ab. Zu beachten ist, dass die ausführbare Datei langsamer startet als die Verzeichnisversion. [38] Der Erstellungsprozess wird mit `pyinstaller -onefile -collect-submodules keras main.py` gestartet. Dabei wird eine sogenannte .spec-Datei erzeugt, die die Spezifikationen enthält. Mit `-onefile` wird angegeben, dass eine einzelne .exe erstellt werden soll. Zusätzlich gibt `-collect-submodules keras` an, dass `KERAS_OCR` erforderlich ist. Neben diesen Optionen können weitere Optionen angegeben werden, die alle in die Spezifikation aufgenommen werden, was auch durch eine manuelle Bearbeitung der Datei .spec geschehen kann. In diesem Fall wurden die folgenden Dateien und Verzeichnisse manuell der .spec-Datei hinzugefügt:

- `ui_labels.json` - Hier befinden sich die in Abschnitt 3.2 erwähnten User Interface (UI)-Labels, die nicht automatisch von PyInstaller erkannt werden.
- `userinterface.qml` - Enthält die Benutzeroberfläche, die in QML entwickelt wurde.
- `icon` - Alle Icons der Anwendung sind hier abgelegt. Dies umfasst die Funktionsbuttons und das Anwendungsicon.
- `Tesseract-OCR` - Die ausführbare Datei enthält auch das Texterkennungswerkzeug `PYTESSERACT`, das in den vorherigen Kapiteln beschrieben wurde.
- `pptx/templates` - Der pyinstaller erkennt die von *python-pptx* mitgebrachten Folien Templates nicht als notwendig an, deswegen müssen diese hinzugefügt werden.
- `argostranslate-models` - Hierbei handelt es sich um einen Satz von Offline-Sprachmodellen, die direkt mit dem Projekt geliefert werden, um eine direkte Ausführung von Übersetzungen in vordefinierte Sprachen auch im Offline-Fall zu ermöglichen.

Anschließend kann die .exe mit dem Befehl `pyinstaller your_spec_file.spec` erstellt und an die Benutzer ausgeliefert werden.

Beim Start der .exe wird ein temporärer Ordner in dem für dieses Betriebssystem vorgesehenen Verzeichnis angelegt. Der Ordner wird nach dem Namensschema _MEIXXXXXX angelegt, wobei XXXXXX eine Zufallszahl ist. [38] Der Vorteil liegt darin, dass mehrere Kopien der Anwendung parallel ausgeführt werden können. Allerdings hätte dies zur Folge, dass eine Menge Speicherplatz in Anspruch genommen werden würde, da die Ressourcen nicht gemeinsam genutzt werden. Nach Beendigung der Anwendung wird diese _MEIXXXXXX gelöscht.

5. Evaluierung des PowerPointDoktors

Das Ziel dieses Kapitels ist es, eine Bewertung des Übersetzungstools PowerPointDoktor zu geben, das im Rahmen dieser Bachelorarbeit entwickelt wurde. Dies wird auf der Grundlage eines im folgenden Kapitel definierten Testumfangs geschehen. Mit dem Testumfang werden die Laufzeiten und die Qualität der Texterkennungswerkzeuge `KERAS_OCR` und `TESSERACT` getestet. Anschließend wird ein praktisches Beispiel näher betrachtet, um zu überprüfen wie der PowerPointDoktor eine PowerPoint-Präsentation bearbeitet, die nicht im Rahmen dieser Bachelorarbeit erstellt wurde. Zum Schluss wird die Laufzeit der Funktion untersucht, die ein PDF-Dokument mit relevanten Folientiteln erstellt.

5.1. Definition des Testumfangs

Der Testumfang beschränkt sich auf den direkten Vergleich der Texterkennungswerkzeuge `KERAS_OCR` und `TESSERACT`. Zu diesem Zweck wurde die `PptxGenerator`-Klasse implementiert. Diese Klasse generiert je nach Bedarf eine beliebige Anzahl von PowerPoint-Präsentationen, die aus einer frei wählbaren Anzahl von Folien bestehen. Jede Folie wird mit einem Titel, einem Textblock und einem generierten Bild versehen. Zur Erzeugung der Bilder wird die durch Belval [11] entwickelte Bibliothek `TextRecognitionDataGenerator` verwendet. Diese Bibliothek bietet zahlreiche Einstellungsmöglichkeiten für die Generierung. So können zum Beispiel Schriftarten, Schriftfarben oder Hintergründe angepasst werden. Darüber hinaus bietet die Bibliothek die Verwendung von vier verschiedenen Text-Generator-Klassen. Für die Generierung werden diese Einstellungen bewusst nicht verändert, um die Zufallsgenerierung nicht zu steuern und eventuell unerkannte Texterkennungsprobleme aufzudecken. Lediglich bei der Wahl der Generator-Klasse wird die `GeneratorFromWikipedia`-Klasse gesetzt. Wie die generierten Textbilder aussehen zeigt die Abbildung 23, welche sechs dieser generierten Textbilder darstellt.

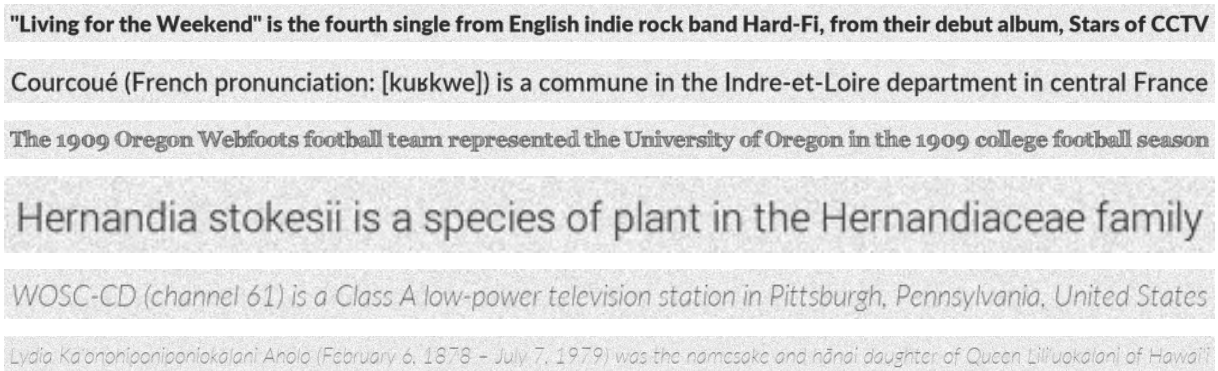


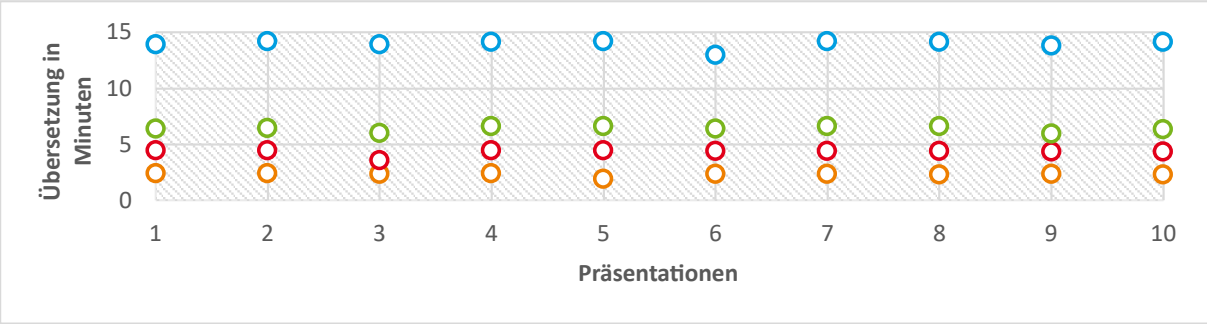
Abbildung 23.: Mit Hilfe der `PptxGenerator`-Klasse generierte Textbilder.

Bei näherer Betrachtung der Bilder fällt auf, dass die erzeugten Bilder unterschiedlich aussehen, selbst wenn für die Farben ausschließlich Graustufen verwendet werden. Die zu untersuchenden Daten sind in diesem Fall jeweils ein Satz von zehn erzeugten PowerPoint-Präsentationen mit jeweils 5, 10, 15 und 30 Bildern pro Präsentation.

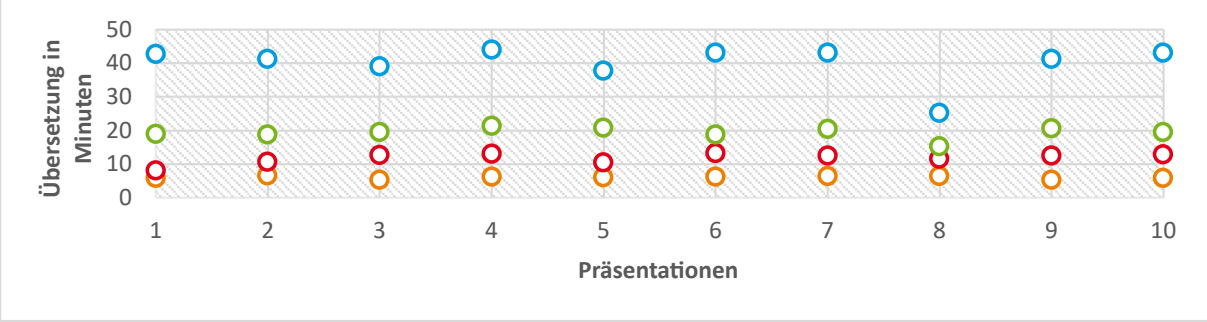
Die Tests wurden auf einem Computer mit einem Sechskern-Prozessor AMD Ryzen 5 2600 mit einer Taktfrequenz von 3,40 GHz und einem Arbeitsspeicher von 16 GB durchgeführt.

5.2. Laufzeit der Präsentations-Übersetzung

Unter Hinzunahme des zuvor definierten Testdatensatzes wird nun die Geschwindigkeit des Übersetzungsprozesses untersucht. Dies geschieht durch den manuellen Start von Übersetzungen für jeweils 10 Präsentationen. Die Präsentationen haben jeweils die gleiche Anzahl an Folienseiten. Zu diesem Zweck wurden die zu übersetzenden Präsentationen im PowerPointDoktor geöffnet und anschließend der Übersetzungsprozess gestartet.



(a) Übersetzungsdauer von 10 verschiedenen PowerPoint-Präsentationen mithilfe von `PYTESSERACT`.



(b) Übersetzungsdauer von 10 verschiedenen PowerPoint-Präsentationen mithilfe von `KERAS_OCR`.

Abbildung 24.: Betrachtung der Übersetzungsdauer unter Nutzung von PowerPoint-Präsentationen mit jeweils 5, 10, 15 oder 30 Bilder.

Die Zeitmessung erfolgt mit Hilfe des Python-Moduls `time` und der Funktion `time()`, die einen Wert in Sekunden zurückgibt, die seit dem 1. Januar 1970 um 00:00:00 (UTC) vergangen sind. Zu Beginn und am Ende der Funktion `start_handling()` der `PowerPointHandler`-Klasse wird diese Funktion zur Berechnung der Differenz und damit der verstrichenen Zeit aufgerufen.

Diese Zeiten sind in Abbildung 24a und Abbildung 24b dargestellt, wobei die orangenen Kreise für alle Präsentationen mit 5 Folien, die roten Kreise für alle Präsentationen mit 10 Folien, die grünen Kreise für alle Präsentationen mit 15 Folien und die blauen Kreise für alle Präsentationen mit 30 Folien steht. Die in Abbildung 24a angegebenen Zeiten beziehen sich auf das `PYTESSERACT` während in Abbildung 24b die Zeiten für `KERAS_OCR` angegeben sind.

In Abbildung 25 sind die Bearbeitungszeiten für zehn Präsentationen mit jeweils 30 Bildern dargestellt. Hier wird unter Berücksichtigung der Gesamtzeit in Sekunden ein Wert für jedes Bild ermittelt. Daraus wurde ein Mittelwert über alle Präsentationen gebildet, der in der letzten Spalte dargestellt ist. Es ist zu erkennen, dass mit `PYTESSERACT` im Mittel nur ca. 28 Sekunden für die Übersetzung benötigt werden. Mit `KERAS_OCR` hingegen werden ca. 80 Sekunden benötigt.

30 Bilder pro Präsentation		PP_1	PP_2	PP_3	PP_4	PP_5	PP_6	PP_7	PP_8	PP_9	PP_10	Durchschnitt
PYTESSERACT	pro Bild in s	27,9	28,5	27,8	28,3	28,4	25,9	28,4	28,3	27,6	28,3	27,9
	Gesamtzeit in s	837,4	854,0	834,5	850,0	852,2	778,2	850,8	848,7	829,5	849,2	838,4
KERAS_OCR	pro Bild in s	85,3	82,5	78,1	87,9	75,4	86,3	86,3	50,5	82,3	86,2	80,1
	Gesamtzeit in s	2558,2	2474,2	2343,2	2637,5	2262,5	2589,7	2589,4	1514,8	2470,0	2586,5	2402,6

Abbildung 25.: Durchlaufzeiten für `PYTESSERACT` und `KERAS_OCR` bei 10 Präsentationen mit jeweils 30 Bildern, gemessen in Sekunden.

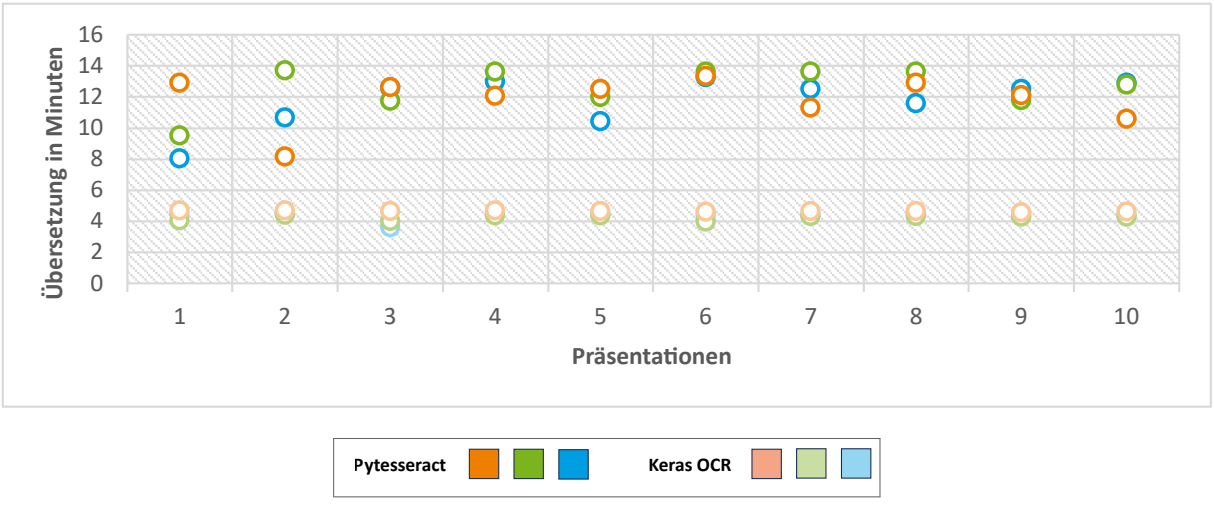


Abbildung 26.: Übersetzungsdauer von 10 verschiedenen PowerPoint-Präsentationen. Hier wurden jeweils drei Durchläufe mit `PYTESSERACT` und `KERAS_OCR` durchgeführt.

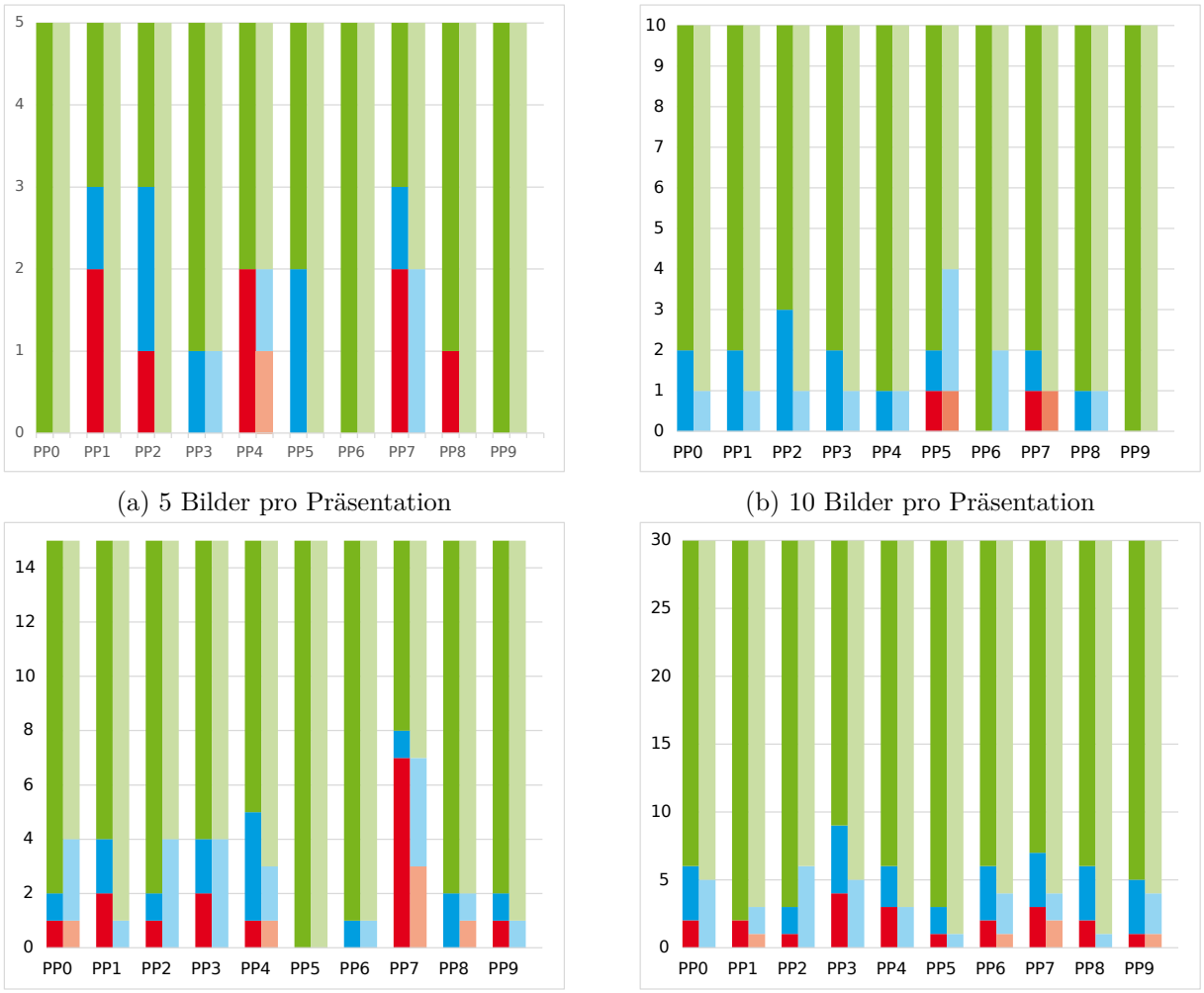
Um zu überprüfen, ob sich die Laufzeit ändert, wenn die selben PowerPoint-Präsentation ein zweites oder drittes Mal ausgeführt werden, wurden für jede Präsentation (10 Bilder) jeweils drei Durchläufe mit den beiden Erkennungswerkzeugen gestartet. Die Resultate sind im Diagramm Abbildung 26 zu sehen. In diesem Diagramm repräsentieren die oberen Kreise die Nutzung von `KERAS_OCR` und die unteren Kreise `PYTESSERACT`.

5.3. Qualität der übersetzen Textbilder

Ziel dieses Unterkapitels ist die Untersuchung der Qualität der generierten Textbilder. Dazu wurden die in Abbildung 23 gezeigten generierten Bilder nach ihrer Erkennung und der anschließenden Textbildgenerierung genauer untersucht. Abbildung 27 zeigt die Ergebnisse für `PYTESSERACT` (oben) und `KERAS_OCR` (unten). Es ist ersichtlich, dass sich die Ergebnisse ähneln, jedoch nicht identisch sind.

Abbildung 27.: Transformierung der in Abbildung 23 gezeigten Textbilder.

Ähnlich wie im vorherigen Unterkapitel, werden in diesem Unterkapitel zehn verschiedene Präsentationen jeweils mit `KERAS_OCR` und `PYTESSERACT` übersetzt. Anschließend wurden diese manuell überprüft, um festzustellen, ob eine Übersetzung stattgefunden hat. Das Ergebnis dieser Überprüfung ist in Abbildung 28 dargestellt. Dabei werden drei Kategorien unterschieden, wobei die dunklen Farben repräsentativ für `PYTESSERACT` und die hellen Farben für `KERAS_OCR` stehen.



(a) 5 Bilder pro Präsentation

(b) 10 Bilder pro Präsentation

(c) 15 Bilder pro Präsentation

(d) 30 Bilder pro Präsentation

Abbildung 28.: Betrachtung der Übersetzungsgenauigkeit.

1. Vollständige Übersetzung: (Grün/hellgrün) - Für inhaltlich korrekte und verständliche Übersetzungen.
2. Fehlerhafte Übersetzung: (Blau/hellblau) - Kennzeichnet inhaltlich abweichende oder nur anteilig korrekte Übersetzungen.
3. Keine Übersetzung: (Rot/hellrot) - Zeigt an, dass keine Übersetzung stattgefunden hat.

An dieser Stelle ist anzumerken, dass es sich bei den PowerPoint-Präsentationen mit 5, 10, 15 und 30 Seiten um unterschiedliche Folien-Seiten handelt und diese nicht auf der Basis der Präsentationen mit 30 Folien reduziert wurden. Es wurden also insgesamt 60 unterschiedliche Folien-Seiten erstellt und hier dargestellt.

Im Folgenden werden die einzelnen Diagramme genauer betrachtet. Bei der Darstellung in Abbildung 28a erreicht `PYTESSERACT` im Durchschnitt für 5 Bilder eine Genauigkeit von 70%. `KERAS_OCR` erreicht hingegen 90%. Wohingegen für `PYTESSERACT` 85% und für `KERAS_OCR` 87% Genauigkeit gezeigt in Abbildung 28b bei 10 Bildern erreicht werden. Diese vollständige Übersetzung ist auch in Abbildung 28c mit 15 Bildern bei 80% für `PYTESSERACT` und bei 82% für `KERAS_OCR` sichtbar. In der letzten Abbildung 28d wird für 30 Bilder mit `PYTESSERACT` eine Genauigkeit von 82,3% und für `KERAS_OCR` von 88% erreicht. Da es sich hier um insgesamt 60 verschiedene Folien mit jeweils einem anderen Bild handelt, kann auch ein Gesamtdurchschnitt berechnet werden, der sich für diese Bilder mit `PYTESSERACT` auf 81,17% und für `KERAS_OCR` auf 86,5% beläuft.

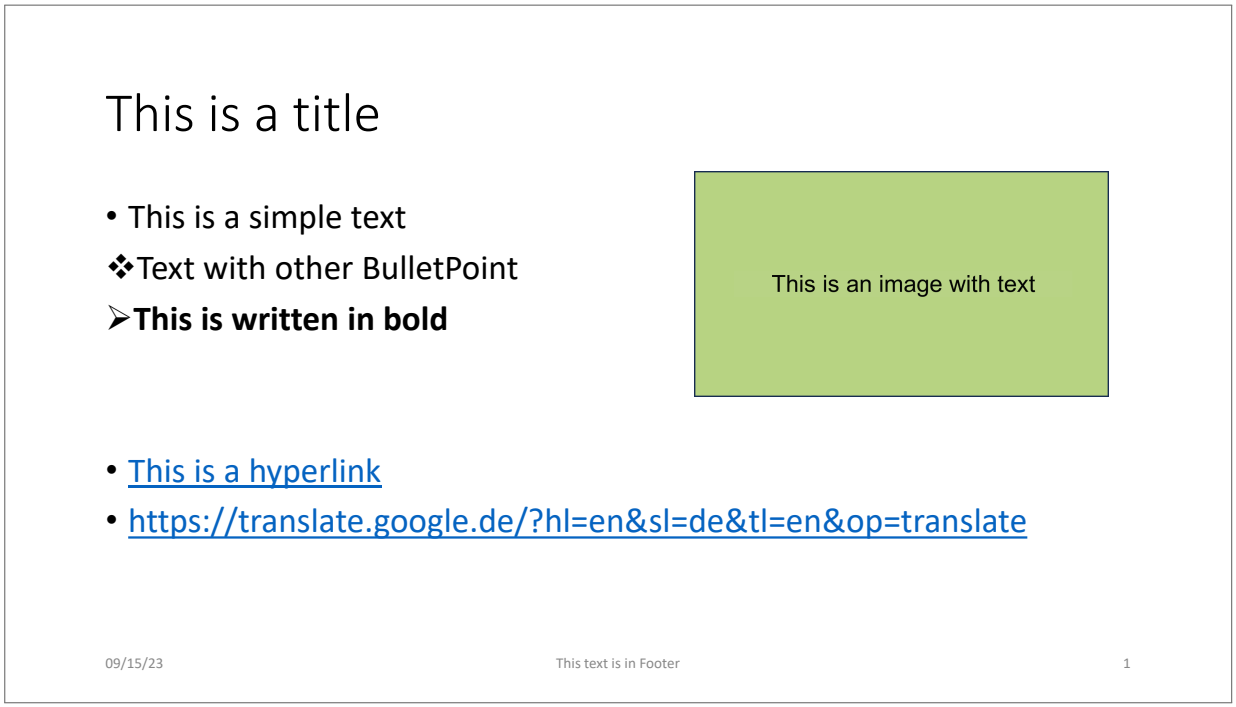


Abbildung 29.: Transformation der Folie aus Abbildung 13.

Betrachtet wird nun ein Beispiel einer Folienübersetzung mit Hilfe von `PYTESSERACT`, die auf der Grundlage der Abbildung 13 aus dem Abschnitt 3.1 erstellt wurde. Aus Darstellungsgründen wurde hier nicht die gesamte PowerPoint-Präsentation aufgenommen, sondern nur die betreffende Folie nach der Übersetzung exportiert und hier dargestellt. Das Ergebnis dieser Übersetzung ist in Abbildung 29 zu sehen. Es zeigt sich, dass sowohl der Text als auch das Textbild unter

Beibehaltung der Stile übersetzt wurden. Weiterhin fällt beim Vergleich des Textbildes auf, dass der übersetzte Bereich kleiner ist als der in Abbildung 13 dargestellte Text. Außerdem ist kaum oder gar nicht zu erkennen, dass dieses Textbild durch den PowerPointDoktor verändert wurde.

5.4. Praxisbeispiel im Detail

Der PowerPointDoktor soll unter anderem in der Lehre unterstützen. Zu diesem Zweck wird die Übersetzung eines Praxisbeispiel betrachtet. Bei der Auswahl der Präsentation wurde darauf geachtet, dass die Präsentation von der Ostfalia - Hochschule für angewandte Wissenschaften stammt und noch einige Bilder und Texte in der deutschen Sprache enthält. Die Wahl fiel hier auf die Infoveranstaltungspräsentation der Fakultät Fahrzeugtechnik aus dem Jahr 2022, die ins Englische übersetzt werden soll. [37]

Auf den 33 Folien dieser Präsentation erkannte der PowerPointDoktor insgesamt 61 Bilder. Diese enthielten insgesamt 481 erkannte Texte, die mit `PYTESSERACT` in 20 Minuten und 39 Sekunden und mit `KERAS_OCR` in 58 Minuten und 41 Sekunde verarbeitet wurden.

Abbildung 30.: Inforveranstaltungspräsentation der Fakultät Fahrzeugtechnik aus dem Jahr 2022. [37, Folie 20]

Genauer betrachtet wird nun die Folie 20 der Infoveranstaltung, gezeigt in Abbildung 30. Darauf sind neben dem Titel zwei Hyperlinks, zwei Bilder, sowie zwei unabhängige Texte zu sehen. Bei dem mittleren Bild handelt es sich um einen Screenshot, der die Vermittlungsplattform

der „Ostfalia für Arbeitsstellen und Praktika“ darstellen soll. Dieses Bild ist wiederum in drei Bereiche unterteilt, die jeweils Texte enthalten. Ganz links befindet sich „Service für Bewerber“, in der Mitte „Service für Unternehmen“ und rechts „Anmelden“, wobei im letzten Fall ein Text - „Ostfalia- Login“ - den darunter liegenden Bereich überdeckt. Dennoch ist erkennbar, dass es sich um eine Anmeldemaske handelt. Diese erlaubt die Eingabe eines „Benutzernamens“ und eines „Passwortes“. Dies kann mit dem Button „Anmelden“ bestätigt werden. Zusätzlich ist ein Schemabild mit dem Text „Allgemeine Stellenportale“ unten rechts in der Abbildung 30 eingefügt.

Abbildung 31.: Mit `PYTESSERACT` transformierte Folie. (Basierend auf: [37, Folie 20])

Die Transformation von Abbildung 30 mit `PYTESSERACT` zeigt Abbildung 31. In diesem Fall wurden alle Texte ins Englische übersetzt, ohne dass die Stile verändert wurden. Das Bild in der Mitte wurde folgendermaßen übersetzt: Aus dem Titeltext „Ostfalia für Arbeitsstellen und Praktika“ wurde „Ostfalia’s placement platform for jobs and internships“. Die Übersetzung der drei Bereiche wurde für den Text links von „Service für Bewerber“ in „Service for applicants“, für den Text in der Mitte von „Service für Unternehmen“ zu „Service for companies“ und für die Maske rechts von „Anmelde“ auf „Register“ vorgenommen. Außerdem wurde der unter dem Textblock „Ostfalia login“ liegende Button „Anmelden“ durch den Text „Register“ ersetzt. Die Qualität dieser Übersetzungen lassen sich alle in die Kategorie der vollständigen Übersetzung einordnen. Das unten rechts liegende Schemabild hingegen ist eine fehlerhafte Übersetzung, da hier aus „Allgemeine Stellenportale“ „General Stellenoortale“ übersetzt wurde.

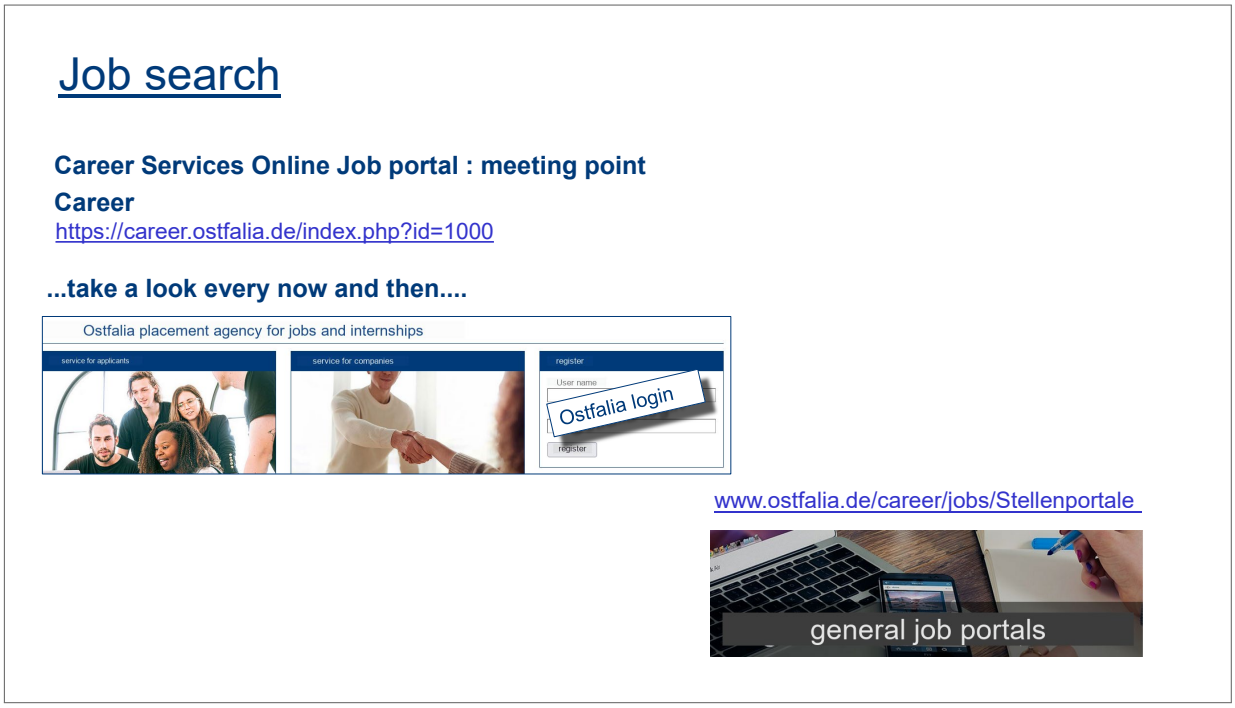


Abbildung 32.: Mit `KERAS_OCR` transformierte Folie. (Basierend auf: [37, Folie 20])

Analog wird die Übersetzung in Abbildung 32 mit `KERAS_OCR` vorgenommen. Da die Text-Elemente identisch übersetzt werden, werden nun die Übersetzungen der Bilder betrachtet. Beim Bild in der Mitte wurde die Überschrift „Ostfalia für Arbeitsstellen und Praktika“ hier abweichend von `PYTESSERACT` mit „Ostfalia placement agency for jobs and internships“ übersetzt. Der Inhalt der drei Bereiche wurde jedoch, bis auf die vollständige Kleinschreibung, identisch übersetzt. Darüber hinaus wurde in der Anmeldemaske „Benutzername“ in „User name“ geändert. Ein weiterer Unterschied zur `PYTESSERACT`-Transformation ist die Texterfassung und Übersetzung des unten rechts platzierten Schemabildes. Dieses wurde in „general job portals“ geändert.

5.5. Laufzeit der PDF-Dokument-Erzeugung

Dieser Abschnitt untersucht die unter Unterabschnitt 3.2.1 implementierte Funktion bezüglich der Erstellung eines PDF-Dokuments, die für den Benutzer relevante Folien-Titel enthält. Dazu wurde die in Abschnitt 5.4 erwähnte Präsentation der Informationsveranstaltung verwendet, die aus 33 Folien besteht. Auf den Folien 1, 16, 19, 22 und 26 wird jeweils ein `MSO_AUTO_SHAPE_TYPE` der Form `CROSS` gesetzt, was über die Eingabemaske in Abbildung 21 auswählbar ist. Nun wird über den vierten Funktionsbutton in Abbildung 18 die PDF-Dokument-Erzeugung gestartet. Auch hier wird die Dauer dieses Vorgangs aufgezeichnet, diese beträgt ca. 4,2 Sekunden. Nun wird die Präsentation erneut geöffnet und ein beliebiger Textblock auf Seite 2 mit der Zeichenkette „===“ eingekapselt. Anschließend wird die PDF-Dokument-Erzeugung erneut gestartet, diesmal dauert der Vorgang 167,9 Sekunden. Das Ergebnis dieser PDF-Dokument-Generierung kann auf Abbildung 33 eingesehen werden. Zum Schluss wird noch ein Durchlauf mit der Suche nach nur einer Zeichenkette gestartet, dieser Durchlauf nimmt 166,7 Sekunden in Anspruch.

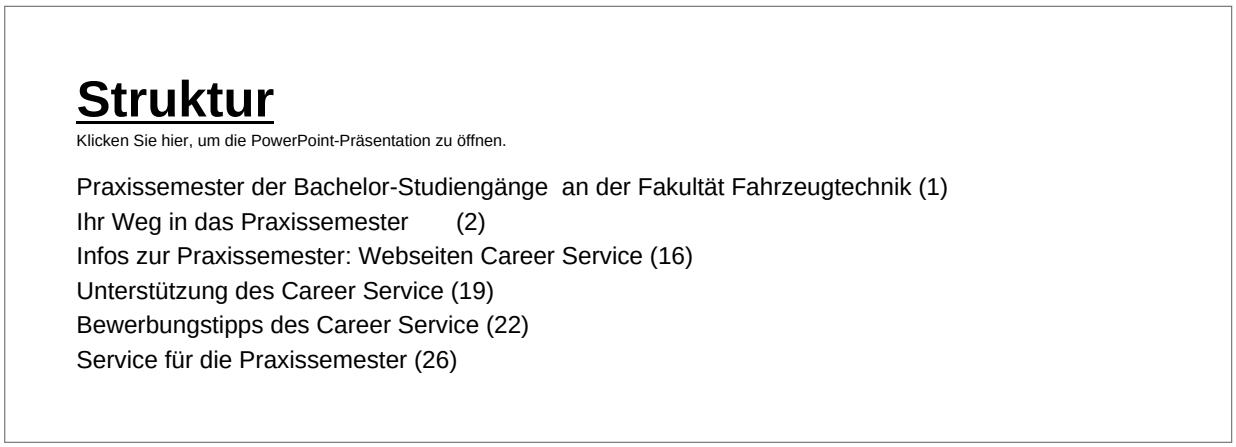


Abbildung 33.: Zugeschnittene PDF-Ausgabe mit relevanten Folien-Titeln.

6. Diskussion

Dieser Abschnitt enthält eine Zusammenfassung und Interpretation der Evaluierungsergebnisse sowie eine Beschreibung aufgetretener Fehler. Darüber hinaus wird eine abschließende Betrachtung der Bachelorarbeit vorgenommen.

6.1. Bewertung der Evaluierung

Von einem Performance-Gesichtspunkt aus betrachtet, zeigt sich, dass `PYTESSERACT` schneller arbeitet im direkten Vergleich mit `KERAS_OCR`.

Bei der Betrachtung der Qualität der Texterkennung zeigt sich, dass die Erkennungsraten beider Systeme ähnlich sind. `PYTESSERACT` ist jedoch besser bei der Erkennung von Sonderzeichen. Wohingegen `KERAS_OCR` diese Eigenschaft generell nicht mit den mitgelieferten trainierten Standardmodellen besitzt, da das entsprechende Netzwerk nur auf Kleinbuchstaben und Zahlen trainiert ist. Hier müsste somit ein eigenständiges Training auf solche Zeichenerkennung erfolgen, was jedoch über den Rahmen der Bachelorarbeit hinausgehen würde. Für das Training kann hier auf `textocr` zurückgegriffen werden, welches fast eine Million beschriftete Bilder mit durchschnittlich 32 Wörtern pro Bild enthält. [46, 53]

Weiterhin fällt bei der Texterkennung auf, dass insbesondere Texte in Bildern mit viel Hintergrundrauschen oder Farbähnlichkeit von Schrift und Hintergrund mit `KERAS_OCR` schlechter abschneiden im Vergleich zu `PYTESSERACT`. Dies ist in Abbildung 34 dargestellt. Hier wurde der Text „lyoio ka ononoononiok agl aholo ebfuoy o 1818 sul tos wcs tne nomcsake ona sono dougner of quccn hfuokoon of owo“ mit `KERAS_OCR` erkannt.

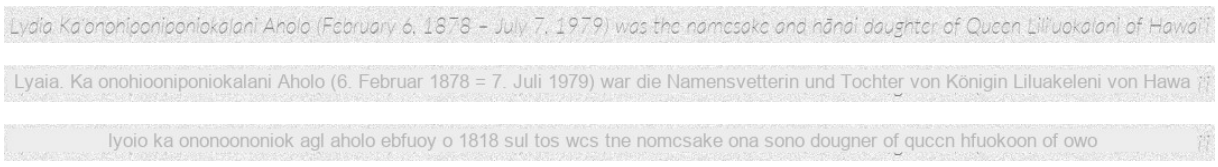


Abbildung 34.: Generiertes Textbild (oben) nach Texterkennung mit `PYTESSERACT` (mitte) und `KERAS_OCR` (unten).

Im Folgenden wird das Praxisbeispiel von Abschnitt 5.4 bewertet. Hier benötigte `PYTESSERACT` etwas mehr als 20 Minuten für die Übersetzung der Texte sowie der Bilder. Wobei die Übersetzung der Bilder, zumindest für die gezeigte 20. Folie, bedeuten würde, hier die entsprechenden Internetseiten aufzurufen, diese unter Verwendung der jeweiligen Zielsprache abzufotografieren und diese dann auf der entsprechenden Folie einzubinden. Dies bedeutet also einen entscheidenden Bearbeitungs- und Geschwindigkeitsvorteil. Bei dem in Abbildung 30 unten rechts dar-

gestellten Schemabild ist eine manuelle Übersetzung unter Umständen gar nicht möglich. Dies könnte daran liegen, dass es dem jeweiligen Benutzer nicht in der Originalversion zur Verfügung steht und somit eine gezielte Neuerstellung beziehungsweise Suche nach einem ähnlichen Bild erforderlich wäre. Weiterhin ist davon auszugehen, dass zumindest bei diesen Folien der Zuhörerkreis anhand der übersetzten Folientexte Rückschlüsse auf die in den Bildern übersetzten Texte ziehen könnte, dies betrifft auch die anteiligen Übersetzungen.

6.2. Aufgetretene Probleme

Bei der Implementierung des PowerPointDoktors sind verschiedene Probleme aufgetreten, von denen die wesentlichen hier behandelt werden. So wurde bei der Untersuchung der PowerPoint-Elemente festgestellt, dass die von der *python-pptx* Bibliothek unterstützten Elemente keine SmartArt-Elemente enthalten. Das Problem liegt laut dem Entwickler der Bibliothek darin, dass das Schema und die Semantik für das Extensible Markup Language (XML) von SmartArt nicht veröffentlicht sind. [47] SmartArt ist jedoch ein wichtiger Bestandteil von PowerPoint-Präsentationen, sodass es laut Microsoft sehr einfach möglich ist, vorhandenen Text in ansprechende Grafiken umzuwandeln. [30] Es wäre denkbar, ähnlich wie beim Ersetzen von rel-paths, diese Texte auf XML-Ebene (Listing 4.1) zu extrahieren und zu übersetzen. Dies würde nicht auf der Grundlage einer Spezifikation erfolgen und käme eher einem Reverse Engineering gleich. Die Alternative wäre die Verwendung der kostenpflichtigen Bibliothek Apose, die laut Application Programming Interface (API)-Eintrag die Bearbeitung von SmartArt-Texten ermöglicht. [6, 7]

Bei der Erstellung der ausführbaren Datei aus Abschnitt 4.4 stellte sich raus, dass die _MEIXXXXXX-Ordner nicht gelöscht werden, wenn die Anwendung unerwartet beendet wird. Ein Grund dafür ist das erzwungene Beenden des PowerPointDoktors durch den Benutzer mit Hilfe des Taskmanagers oder der unter anderen Betriebssystemen üblichen Methoden. Dies gilt auch, wenn das Programm abstürzt.

Eine weitere Besonderheit ist die Erstellung von Textbildern. Betrachtet wird dazu Abbildung 30, das verschiedene Bilder zeigt, darunter einen Screenshot mit Schatten, der aber bei der Textbildgenerierung in Abbildung 31 nicht übernommen wird. Der Grund dafür ist, dass die *python-pptx* Bibliothek es nicht erlaubt, einen Schatten zu setzen, beziehungsweise zu verändern. Es ist lediglich möglich, den Schatten zu deaktivieren. [48]

Bei der Auswertung von Abschnitt 5.5 hat sich herausgestellt, dass die Wahl der `Shapes` bewusst so gewählt werden sollte, dass diese nicht bereits in den Folien gesetzt sind. Speziell im Fall des Praxisbeispiels aus Abschnitt 5.4 wurden rote Rechtecke verwendet, um einzelne Bilder hervorzuheben, daher wurde bei der Auswahl der `Shapes` auf das in Abschnitt 4.2 beschriebene `CROSS` zurückgegriffen. Weiterhin ist es möglich, dass ein `Auto_Shape` ein Hintergrundbild hat, das vom PowerPointDoktor korrekt erkannt wird, aber bei der PDF-Dokument-Erzeugung

fälschlicherweise als relevante Folie erkannt wird. [16]

Zeilenumbrüche in übersetzten Texten, die dazu führen, dass ein generiertes Textbild scheinbar leer ist. Durch die Zeilenumbrüche wurden die Texte jedoch außerhalb des sichtbaren Bereichs dargestellt. Am Beispiel des dritten Textes in Abbildung 23 wurde so aus „The 1909 Oregon Webfoots football team represented the University“ nach der Übersetzung „Die Fußballmannschaft Oregon Webfoots aus dem Jahr 1909 vertrat die Universität\n\nvon Oregon in der College-Football-Saison 1909“.

6.3. Zusammenfassung der Arbeit

Ziel dieser Bachelorarbeit war die Konzeption und Implementierung des PowerPointDoktors, einer Anwendung, die in der Lage ist, einen bestimmten Personenkreis bei der Übersetzung, Verwaltung und Transformation von PowerPoint-Präsentationen zu unterstützen.

Umgesetzt wurde dies mit Python als Programmiersprache und PySide6 für die Frontend-Gestaltung. Die in der Benutzeroberfläche auswählbaren Grundfunktionen bieten den Anwendern eine zentrale Möglichkeit, PowerPoint-Präsentationen umzubenennen, diese zu duplizieren, den Ordner, in dem sich die Präsentation befindet, zu öffnen oder die PowerPoint-Präsentation aus dem PowerPointDoktor heraus zu starten. Darüber hinaus gibt es Funktionen, die mit der Hilfe von der *python-pptx* Bibliothek umgesetzt wurden, wie das Ersetzen von absoluten Dateipfaden durch relative Dateipfade, das Erstellen eines PDF-Dokuments, das die relevanten Folien Titel enthält, sowie das Übersetzen einer PowerPoint-Präsentation.

Für die Erkennung der Texte innerhalb der in den PowerPoint-Präsentationen gefundenen Bilder wurden die beiden Texterkennungswerkzeuge `PYTESSERACT` und `KERAS_OCR` integriert. Rückblickend wäre es für die Implementierung hilfreich gewesen, die beiden Texterkennungswerkzeuge mit ihren Stärken und Schwächen auf der Grundlage von bestehenden Arbeiten zu untersuchen.

Um die gefundenen Texte zu übersetzen wurden die Python-Bibliotheken `googletrans` und `argostranslate` verwendet. Zusätzlich wurde im zweiten Tab des PowerPointDoktors eine zentrale Oberfläche zum Download der Offline-Sprachmodelle von `argostranslate` erstellt.

Der PowerPointDoktor ist in der Lage alle von der *python-pptx* Bibliothek unterstützen Text-Shapes zu übersetzten. Außerdem berücksichtigt diese Anwendung bei der Übersetzung von Bildern horizontal ausgerichtete Texte. Bei dem Erstellen der übersetzen Bilder wird weiterhin mit Hilfe des KMeans Algorithmus eine ähnliche Farbe für die Hintergrund-, sowie Schriftfarbe berechnet. Hiermit wird versucht eine natürliche Einblendung beim Ersetzen der Textbereiche in den Bildern zu erzielen.

Die Implementierung des PowerPointDoktors hat einen gewissen Stand erreicht, der jedoch durch weitere Ansätze und Ideen perspektivisch erweitert werden kann.

7. Ausblick

Dieses Kapitel gibt einen Überblick darüber, in welchen Bereichen der PowerPointDoktor noch erweitert werden könnte. Es werden weitere Funktionen vorgestellt, die die Qualität der Übersetzungen oder die Qualität der Anwendung in Bezug auf Benutzerfreundlichkeit und Leistung verbessern.

7.1. Übersetzungsqualität

Zur Zeit werden Texte vom PowerPointDoktor nur erkannt, wenn sie horizontal dargestellt sind. Es ist jedoch möglich, dass der Text um 90° oder 270° gedreht oder in einem anderen Winkel dazwischen dargestellt ist. Es wäre daher denkbar, die **Texterkennung um einen Rotationswert zu erweitern** und diesen bei der Textbilderstellung zu berücksichtigen, um so bei der Bilderstellung und beim Einfügen die ursprüngliche Darstellung zu erhalten, dadurch könnte der von Akhter und Rege [3] beschriebene Ansatz in Betracht gezogen werden.

Interessant wäre auch eine Erweiterung der Textbildgenerierung in Unterunterabschnitt 4.3 um die **Erkennung der Schriftart**. Dies könnte mit Hilfe des in der Arbeit von Wang et al. [60] und Reni [45] vorgestellten DeepFont-Systems realisiert werden. Auf diese Weise könnte eine Verbesserung der Bildqualität durch die Integration eines zum Originalbild passenden Textstils erreicht werden. Dabei muss es sich nicht um den identischen Schriftstil handeln, aber ein ähnlicher Stil würde zu einer noch nahtloseren Integration der Textbilder führen.

Der Schwerpunkt dieser Arbeit liegt auf der Erkennung von Text in Bildern, dies funktioniert nicht immer fehlerfrei und neben der Aufbereitung der Bilder könnte auch eine Nachbearbeitung der erkannten Texte erfolgen. Dazu könnte eine Überprüfung mittels **Wörterbuch oder einer Rechtschreibprüfung** genutzt werden. Somit könnten Fehler, wie „Stellenoortal“ in Abbildung 31 gefunden und korrigiert werden. [10]

Leider gibt es keine **Unterstützung für SmartArt-Objekte** in der *python-pptx* Bibliothek. Aus Abschnitt 2.5 ist deutlich geworden, dass theoretisch jede Präsentation auf Archivebene verändert werden kann. Es sollte also möglich sein, die Präsentationen unabhängig von der verwendeten Bibliothek anzupassen.

7.2. Benutzerfreundlichkeit und -feedback

Der **Fortschritt der Übersetzung** könnte durch die Anzahl der übersetzten Folien angezeigt werden. Zusätzlich könnten die in Übersetzung befindlichen Elemente wie Text, Bild oder Tabelle angezeigt werden, um den aktuellen Status und die Dauer der Übersetzung zu verdeutlichen.

Eine Möglichkeit wäre es, **PowerPoint-Präsentationen direkt per Drag & Drop in die Anwendung** zu importieren. Dies würde eine schnelle Konvertierung von Präsentationen ermöglichen, ohne den Ordner auswählen zu müssen.

Derzeit werden alle Präsentationen in einer gemeinsamen Liste angezeigt, die in dem ausgewählten Ordner und seinen Unterordnern gefunden wurden. Dies ist zwar sehr übersichtlich und einheitlich, kann aber ab einer bestimmten Anzahl von Präsentationen zu einer unübersichtlichen Darstellung führen. Denkbar ist daher die Benutzeroberfläche in Abbildung 18 um eine **Suchmaske zu ergänzen**.

In den in Kapitel 5 gezeigten Beispielen war die Sprache immer vorher bekannt. Wenn der Benutzer jedoch nicht weiß, welche Sprache in der jeweiligen Präsentation verwendet wird oder wenn es sich um eine Sprache handelt, die der Benutzer nicht kennt, können Probleme auftreten. Eine sinnvolle Erweiterung wäre daher, wenn der PowerPointDoktor die **Sprache automatisch erkennen** und die Ausgangssprache selbstständig einstellen könnte. Eine alternative Lösung wäre es über einen weiteren Button die Spracherkennung auszulösen.

7.3. Verbesserung der Leistungsfähigkeit und Effizienz der Anwendung

Das Problem, dass die **temporären _MEIXXXXXX Ordner nicht gelöscht** werden, wenn die Anwendung unerwartet beendet wird. Es wäre empfehlenswert, eine Routine zu schreiben, die diese temporären Ordner löscht. Eine Möglichkeit besteht darin beim Start von PowerPointDoktor eine Datei mit der Prozess-ID in den Ordner _MEIXXXXXX zu schreiben. Die Anwendung würde dann bei jedem Start alle existierenden _MEIXXXXXX Ordner öffnen und die Prozess-ID aus der Datei lesen. Dann würde sie die Prozesse, die gerade ausgeführt werden, mit dieser ID vergleichen. Wenn es keine Übereinstimmung gibt, würde es den entsprechenden _MEIXXXXXX-Ordner löschen und so wieder Speicherplatz auf dem Computer freizugeben.

Wie im Testkapitel gezeigt, ist die Übersetzung von Präsentationen, die Bilder enthalten, sehr zeitaufwendig. Der Grund dafür ist die imperative Implementierung, da jedes Bild einzeln nacheinander übersetzt wird, was in Summe sehr viel Zeit in Anspruch nimmt. Hier könnte ein Parallelisierungsansatz in Betracht gezogen werden. Denkbar wäre die **parallele Übersetzung** aller Folien und/oder die parallele Übersetzung aller Textbereiche eines erkannten Bildes. Gerade letzteres könnte eine enorme Zeitersparnis bringen.

Derzeit muss jede Übersetzung einzeln angestoßen werden. Für den Fall, dass der Nutzer **mehrere Präsentationen gleichzeitig übersetzen** oder sie direkt in mehrere Zielsprachen übersetzen möchte. Zumindest im letzteren Fall würde dies bei der derzeitigen Umsetzung einen Zeitgewinn im Vergleich zu einer einzelnen Übersetzung mit sich bringen.

Literaturverzeichnis

[1] Ajith Abraham. „Artificial Neural Networks“. In: *Handbook of Measuring System Design*. John Wiley & Sons, Ltd, 2005. Kap. 129. ISBN: 9780471497394. DOI: https://doi.org/10.1002/0471497398.mm421. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/0471497398.mm421.

[2] S Agatonovic-Kustrin und R Beresford. „Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research“. In: *Journal of Pharmaceutical and Biomedical Analysis* 22.5 (2000), S. 717–727. ISSN: 0731-7085. DOI: https://doi.org/10.1016/S0731-7085(99)00272-1. URL: https://www.sciencedirect.com/science/article/pii/S0731708599002721.

[3] Shaheera Saba Mohd Naseem Akhter und Priti P Rege. „Improving Skew Detection and Correction in Different Document Images Using a Deep Learning Approach“. In: *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. 2020, S. 1–6. DOI: 10.1109/ICCCNT49239.2020.9225619.

[4] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Q. Al-Dujaili, Ye Duan, Omran Al-Shamma, José I. Santamaría, Mohammed Abdulraheem Fadhel, Muthana Al-Amidie und Laith Farhan. „Review of deep learning: concepts, CNN architectures, challenges, applications, future directions“. In: *Journal of Big Data* 8 (2021). URL: https://api.semanticscholar.org/CorpusID:232434552.

[5] Yali Amit und Pedro Felzenszwalb. „Object Detection“. In: *Computer Vision: A Reference Guide*. Hrsg. von Katsushi Ikeuchi. Boston, MA: Springer US, 2014, S. 537–542. ISBN: 978-0-387-31439-6. DOI: 10.1007/978-0-387-31439-6_660. URL: https://doi.org/10.1007/978-0-387-31439-6_660.

[6] Aspose. *Python PowerPoint API for Presentations. Python PPTX, PPT*. 2023. URL: https://products.aspose.com/slides/python-net/ (besucht am 07. 11. 2023).

[7] Aspose. *Replace text in smart art*. 2023. URL: https://docs.aspose.com/cells/net/replace-text-in-smart-art/ (besucht am 07. 11. 2023).

[8] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun und Hwalsuk Lee. *Character Region Awareness for Text Detection*. 2019. arXiv: 1904.01941 [cs.CV].

[9] Dzmitry Bahdanau, Kyunghyun Cho und Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

[10] Tyler Barrus. *pyspellchecker Documentation*. 2023. URL: https://readthedocs.org/projects/pyspellchecker/downloads/pdf/latest/ (besucht am 12. 11. 2023).

[11] Edouard Belval. *TextRecognitionDataGenerator's documentation*. 2019. URL: https://textrecognitiondatagenerator.readthedocs.io/en/latest/index.html (besucht am 05. 11. 2023).

[12] Alexei Botchkarev. „A New Typology Design of Performance Metrics to Measure Errors in Machine Learning Regression Algorithms". In: *Interdisciplinary Journal of Information, Knowledge, and Management* 14 (2019). DOI: 10.28945/4184. URL: https://doi.org/10.28945%2F4184.

[13] Syed Saqib Bukhari, Faisal Shafait und Thomas M. Breuel. „Text-Line Extraction Using a Convolution of Isotropic Gaussian Filter with a Set of Line Filters". In: *2011 International Conference on Document Analysis and Recognition*. 2011, S. 579–583. DOI: 10.1109/ICDAR.2011.122.

[14] Steve Canny. *Image*. 2013. URL: https://python-pptx.readthedocs.io/en/latest/api/image.html (besucht am 25. 10. 2023).

[15] Steve Canny. *MSO_SHAPE_TYPE*. 2013. URL: https://python-pptx.readthedocs.io/en/latest/api/enum/MsoShapeType.html?highlight=mso_shape_type (besucht am 25. 10. 2023).

[16] Steve Canny. *Understanding Shapes*. 2013. URL: https://python-pptx.readthedocs.io/en/latest/user/understanding-shapes.html (besucht am 23. 10. 2023).

[17] K. Choo, E. Greplova, M.H. Fischer und T. Neupert. *Machine Learning kompakt: Ein Einstieg für Studierende der Naturwissenschaften*. essentials. Springer Fachmedien Wiesbaden, 2021. ISBN: 9783658322670. URL: https://doi.org/10.1007/978-3-658-32268-7.

[18] Vincent Dumoulin und Francesco Visin. *A guide to convolution arithmetic for deep learning*. 2018. arXiv: 1603.07285 [stat.ML].

[19] ecma. *ECMA-376*. Dez. 2021. URL: https://www.ecma-international.org/publications-and-standards/standards/ecma-376/ (besucht am 25. 09. 2023).

[20] googletrans. *Googletrans: Free and Unlimited Google translate API for Python*. 2020. URL: https://py-googletrans.readthedocs.io/en/latest/ (besucht am 26. 10. 2023).

[21] Frauke Günther und Stefan Fritsch. „neuralnet: Training of Neural Networks". In: *The R Journal* 2.1 (2010), S. 30. DOI: 10.32614/rj-2010-006. URL: https://doi.org/10.32614/rj-2010-006.

[22] Abdeslam Harraj und Naoufal Raissouni. „OCR Accuracy Improvement on Document Images Through a Novel Pre-Processing Approach". In: *Signal & Image Processing : An International Journal* 6 (Sep. 2015). DOI: 10.5121/sipij.2015.6401.

[23] Innofact. *Umfrage zur Nutzungshäufigkeit von PowerPoint zur Präsentationserstellung 2018*. Juni 2018. URL: https://de.statista.com/statistik/daten/studie/879656/umfrage/umfrage-zur-nutzungshaeufigkeit-von-powerpoint-zur-praesentationserstellung/ (besucht am 18. 09. 2023).

[24] Innofact. *Umfrage zur Nutzungshäufigkeit von Programmen zur Präsentationserstellung 2018*. Juni 2018. URL: https://de.statista.com/statistik/daten/studie/879452/umfrage/umfrage-zur-nutzungshaeufigkeit-von-programmen-zur-praesentationserstellung/ (besucht am 18. 09. 2023).

[25] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart und Alexander M. Rush. *OpenNMT: Open-Source Toolkit for Neural Machine Translation*. 2017. arXiv: 1701.02810 [cs.CL].

[26] Steven W. Knox. *Machine Learning: a Concise Introduction*. Wiley, März 2018. DOI: 10.1002/9781119439868. URL: https://doi.org/10.1002/9781119439868.

[27] Jonathan Long, Evan Shelhamer und Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2015. arXiv: 1411.4038 [cs.CV].

[28] A. Phillips M. Davis. *Tags for the Identification of Languages*. Sep. 2009. URL: https://www.ietf.org/rfc/rfc5646.txt (besucht am 16. 10. 2023).

[29] Microsoft. *[MS-OE376]: Office Implementation Information for ECMA-376 Standards Support*. Aug. 2022. URL: https://learn.microsoft.com/en-us/openspecs/office_standards/ms-oe376/db9b9b72-b10b-4e7e-844c-09f88c972219 (besucht am 25. 09. 2023).

[30] Microsoft. *Create a SmartArt graphic from scratch*. 2020. URL: https://support.microsoft.com/en-gb/office/create-a-smartart-graphic-from-a-list-in-powerpoint-ed299a87-43e2-4a18-a3ba-cc90c2149e33 (besucht am 04. 11. 2023).

[31] Microsoft. *Extract files or objects from a PowerPoint file*. 2023. URL: https://support.microsoft.com/en-au/office/extract-files-or-objects-from-a-powerpoint-file-85511e6f-9e76-41ad-8424-eab8a5bbc517 (besucht am 25. 09. 2023).

[32] Microsoft. *Insert and play a video file from your computer*. 2023. URL: https://support.microsoft.com/en-gb/office/insert-and-play-a-video-file-from-your-computer-f3fcbd3e-5f86-4320-8aea-31bff480ed02#:~:text=On%20the%20Insert%20tab%2C%20in,arrow%2C%20and%20then%20click%20Insert (besucht am 25. 09. 2023).

[33] Microsoft. *MsoAutoShapeType enumeration (Office)*. 2023. URL: https://learn.microsoft.com/en-us/office/vba/api/office.msoautoshapetype (besucht am 23. 10. 2023).

[34] Hyeonwoo Noh, Seunghoon Hong und Bohyung Han. *Learning Deconvolution Network for Semantic Segmentation*. 2015. arXiv: 1505.04366 [cs.CV].

[35] Keiron O'Shea und Ryan Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE].

[36] *PIL how to scale text size in relation to the size of the image*. 2011. URL: https://stackoverflow.com/a/61891053 (besucht am 21. 10. 2023).

[37] Kornelia Pusch Prof. Dr.-Ing. Köhring. *Praxissemster-Infoveranstaltung*. 2022. URL: https://www.ostfalia.de/cms/de/career/praxis/fakultaet-f/infoveranstaltung/Praesentation_Fahrzeugtechnik_2022_11.pptx (besucht am 29. 10. 2023).

[38] PyInstaller. *What PyInstaller Does and How It Does It*. 2023. URL: https://pyinstaller.org/en/stable/operating-mode.html#bundling-to-one-file (besucht am 23. 10. 2023).

[39] python. *GUI Programming in Python*. 2023. URL: https://wiki.python.org/moin/GuiProgramming (besucht am 30. 09. 2023).

[40] Qt. *Material Design*. 2023. URL: https://doc.qt.io/qt-6/qtquickcontrols-material.html#detailed-desc-material (besucht am 18. 10. 2023).

[41] Qt. *QML Applications*. 2023. URL: https://doc.qt.io/qtforpython-6/overviews/qmlapplications.html#qml-applications (besucht am 15. 10. 2023).

[42] Qt. *Qt Designer Manual*. 2023. URL: https://doc.qt.io/qt-6/qtdesigner-manual.html (besucht am 03. 10. 2023).

[43] Qt. *Qt Namespace*. 2023. URL: https://doc.qt.io/qt-6/qt.html#ItemFlag-enum (besucht am 19. 10. 2023).

[44] Gopinath Rebala, Ajay Ravi und Sanjay Churiwala. „Machine Learning Definition and Basics". In: *An Introduction to Machine Learning*. Cham: Springer International Publishing, 2019, S. 1–17. ISBN: 978-3-030-15729-6. DOI: 10.1007/978-3-030-15729-6_1. URL: https://doi.org/10.1007/978-3-030-15729-6_1.

[45] Robin Reni. *Font_Recognition-DeepFont*. 2021. URL: https://github.com/robinreni96/Font_Recognition-DeepFont (besucht am 12. 11. 2023).

[46] Adrian Rosebrock. *Improving OCR Results with Basic Image Processing*. Nov. 2021. URL: https://pyimagesearch.com/2021/11/22/improving-ocr-results-with-basic-image-processing/ (besucht am 02. 11. 2023).

[47] Scanny. *How to read PowerPoint SmartArt Data using python ? Python-PPTX*. Apr. 2020. URL: https://stackoverflow.com/a/61016223 (besucht am 23. 10. 2023).

[48] Scanny. *Shadow*. Apr. 2013. URL: https://python-pptx.readthedocs.io/en/latest/dev/analysis/shp-shadow.html#protocol (besucht am 23. 10. 2023).

[49] Sagar Sharma. *What the Hell is Perceptron? - The Fundamentals of Neural Networks*. 2017. URL: https://towardsdatascience.com/what-the-hell-isperceptron-%20626217814f53 (besucht am 20. 09. 2023).

[50] Siddharth Sharma, Simone Sharma und Anidhya Athaiya. „ACTIVATION FUNCTIONS IN NEURAL NETWORKS“. In: *International Journal of Engineering Applied Sciences and Technology* (2020). URL: https://api.semanticscholar.org/CorpusID:225922639.

[51] Evan Shelhamer, Jonathan Long und Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2016. arXiv: 1605.06211 [cs.CV].

[52] Baoguang Shi, Xiang Bai und Cong Yao. *An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition*. 2015. arXiv: 1507.05717 [cs.CV].

[53] Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach und Amanpreet Singh. *TextCaps: a Dataset for Image Captioning with Reading Comprehension*. 2020. arXiv: 2003.12462 [cs.CV].

[54] R. Smith. „An Overview of the Tesseract OCR Engine“. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Bd. 2. 2007, S. 629–633. DOI: 10.1109/ICDAR.2007.4376991.

[55] Statista. *Künstliche Intelligenz: Detaillierte Marktanalyse*. Apr. 2023. URL: https://de.statista.com/statistik/studie/id/50489/dokument/in-depth-report-artificial-intelligence/ (besucht am 18. 09. 2023).

[56] Statista. *Sprachübersetzung NLP - Weltweit*. Aug. 2023. URL: https://de.statista.com/outlook/tmo/kuenstliche-intelligenz/natural-language-processing/sprachuebersetzung-nlp/weltweit#wert (besucht am 18. 09. 2023).

[57] Mohammad Mustafa Taye. „Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions“. In: *Computation* 11.3 (2023). ISSN: 2079-3197. DOI: 10.3390/computation11030052. URL: https://www.mdpi.com/2079-3197/11/3/52.

[58] Argos Open Technologies. *Argos Translate Documentation*. 2020. URL: https://argos-translate.readthedocs.io/en/latest/ (besucht am 26. 10. 2023).

[59] Paul Voigtlaender, Jonathon Luiten, Philip H. S. Torr und Bastian Leibe. *Siam R-CNN: Visual Tracking by Re-Detection*. 2020. arXiv: 1911.12836 [cs.CV].

[60] Zhangyang Wang, Jianchao Yang, Hailin Jin, Eli Shechtman, Aseem Agarwala, Jonathan Brandt und Thomas S. Huang. *DeepFont: Identify Your Font from An Image*. 2015. arXiv: 1507.03196 [cs.CV].

[61] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes und Jeffrey Dean. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. 2016. arXiv: 1609.08144 [cs.CL].

[62] Rikiya Yamashita, Mizuho Nishio, Richard Do und Kaori Togashi. „Convolutional neural networks: an overview and application in radiology". In: *Insights into Imaging* 9 (Juni 2018). DOI: 10.1007/s13244-018-0639-9.

[63] Mengmeng Zhang, Wei Li und Qian Du. „Diverse Region-Based CNN for Hyperspectral Image Classification". In: *IEEE Transactions on Image Processing* 27.6 (2018), S. 2623–2634. DOI: 10.1109/TIP.2018.2809606.

[64] Zhilu Zhang und Mert R. Sabuncu. *Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels*. 2018. arXiv: 1805.07836 [cs.LG].

[65] Alan Zucconi. *The Secrets of Colour Interpolation*. Juni 2016. URL: https://www.alanzucconi.com/2016/01/06/colour-interpolation/ (besucht am 15. 10. 2023).

Appendix

Anhang A:

Betriebshandbuch

Im Folgenden wird beschrieben, wie die Entwicklungsumgebung für den PowerPointDoktor installiert werden kann. Es folgt eine Beschreibung wie die Anwendung gestartet und benutzt werden kann. Abschließend wird beschrieben, wie die ausführbare Datei erstellt werden kann.

A.1: Einrichtung der Entwicklungsumgebung

Der PowerPointDoktor wurde unter **Windows 10** mit der **Python Version 3.9.13** entwickelt. Um den PowerPointDoktor weiterzuentwickeln, muss die Entwicklungsumgebung dafür eingerichtet werden. Hierfür muss ein Zugang zum folgenden Git-Repository vorhanden sein:

- **Link:** `https://github.com/DirkJLehmann/UebersetzenVonTextenInPraesentationen`
- **Besitzer:** Prof. Dr.-Ing. habil. Dirk Joachim Lehmann
- **Kontakt:** `mailto:di.lehmann@ostfalia.de`

Sobald der Zugriff gewährt wurde, muss das Git-Repository auf das lokale Entwicklungssystem geklont werden. Danach muss in das soeben geklonte Repository im Ordner `/UebersetzenVonTextenInPraesentationen` gewechselt werden. Hier müssen mit dem Konsolen-Befehl *pip install -r requirements.txt* die für die Entwicklung benötigten Bibliotheken installiert werden. Der Inhalt von `requirements.txt` kann unter Listing A.1 eingesehen werden. Zur Vereinfachung der Entwicklung wird die Verwendung einer Integrated Development Environment (IDE) empfohlen.

Nützliche Links:

- Python Downloads - `https://www.python.org/downloads/`
- Github Dokumentation - `https://docs.github.com/de/repositories/creating-and-managing-repositories/cloning-a-repository`
- Python IDEs - `https://wiki.python.org/moin/IntegratedDevelopmentEnvironments`

```
# picture handling
numpy == 1.21.6
tensorflow == 2.11.0
keras-ocr == 0.9.2
torchvision == 0.14.1
craft_text_detector == 0.4.3
scikit-learn == 1.2.1

# pptx files
python-pptx >= 0.6.21
fpdf2 == 2.7.1

# user interface
pyside6 == 6.4.2

# translation lib
requests == 2.28.2
googletrans == 3.1.0a0
argostranslate == 1.8.0

pyinstaller >= 5.13.1
```

Listing A.1: In der `requirements.txt` definierte Bibliotheken.

A.2: Starten und Nutzen des Tools

Es gibt zwei Möglichkeiten, den PowerPointDoktor zu starten. Variante 1 ist, in den `/src` Ordner des `UebersetzenVonTextenInPraesentationen` Repositories zu wechseln und von der Konsole aus den Befehl *python main.py* einzugeben. Nach kurzer Zeit sollte das PowerPointDoktor-Fenster sichtbar sein. Die zweite Möglichkeit ist über die ausführbare Datei. Sollte keine ausführbare Datei vorhanden sein, kann diese wie unter Abschnitt A.3 beschrieben erstellt werden.

Nach dem Start des PowerPointDoktors öffnet sich der PowerPoint Scanner (1), dargestellt in Abbildung 35. Hier ist die Listenansicht (17) zunächst leer. Durch Anklicken des Buttons (5) öffnet sich der Windows Explorer und es kann ein Ordner ausgewählt und mit `[Ordner auswählen]` bestätigt werden. Wenn sich in diesem Ordner und seinen Unterordnern PowerPoint-Präsentationen befinden, werden diese in der Liste (17) angezeigt. Sollte sich der Inhalt dieser Liste außerhalb des PowerPointDoktors geändert haben, so kann diese mit einem Klick auf den Button (4) aktualisiert werden.

Alle aufgelisteten PowerPoint-Präsentationen können durch Klicken auf den Namen einer PowerPoint-Präsentation (7) geändert werden. Wichtig: Erst nach Bestätigung mit der Enter-Taste wird die Änderung übernommen!

Die Buttons (8) bis (12) haben folgende Funktionen:

- Der Button (8) öffnet den Ordner, in dem die betrachtete PowerPoint-Präsentation liegt und selektiert diese.
- Der Button (9) erstellt eine Kopie der betreffenden PowerPoint-Präsentation. Der Name der Kopie ist der ursprüngliche Name mit der Endung „_copy“.
- Über den Button (10) wird die entsprechende PowerPoint-Präsentation mit Microsoft PowerPoint geöffnet.
- Über den Button (11) wird ein PDF-Dokument mit den relevanten Überschriften aus der PowerPoint-Präsentation erstellt. Hinweis: Hierfür müssen über den Konfiguration-Reiter (3) (Abbildung 37) Einstellungen vorgenommen werden.
- Über den Button (12) wird eine neue PowerPoint-Präsentation erstellt bei der alle eingefügten Dateien mit ihren absoluten Pfaden durch relative Dateipfade ersetzt wurden.

Das Übersetzen der PowerPoint-Präsentationen erfolgt nach Auswahl der Quellsprache (13) und Zielsprache (15) über den Button (16). Um die Quell- und Zielsprachen schnell zu tauschen kann der Button (14) verwendet werden.

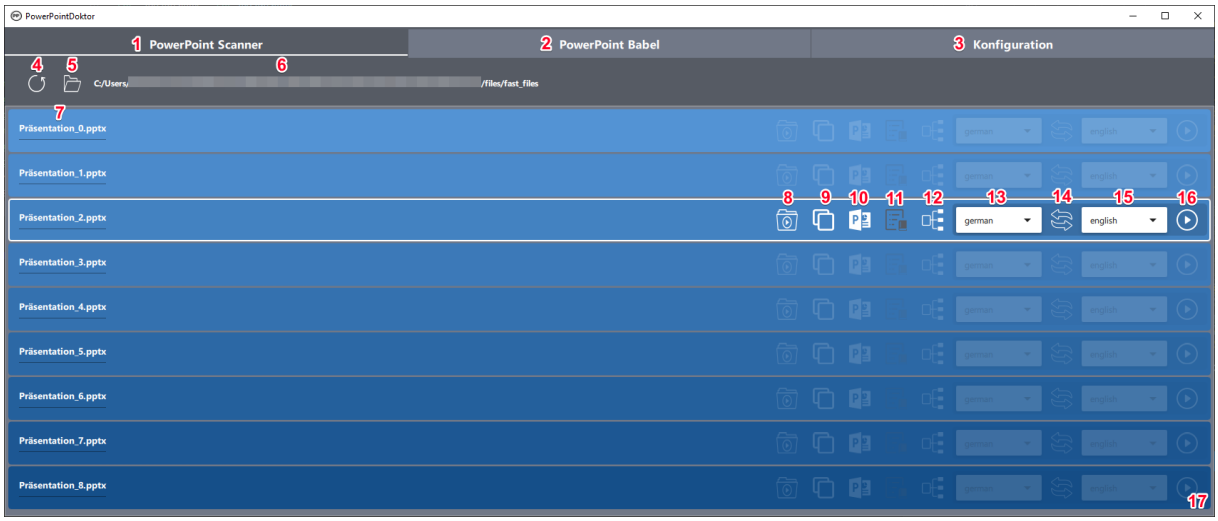


Abbildung 35:: PowerPoint Scanner - Listenansicht der PowerPoint-Präsentationen.

Der Online-Übersetzungsdienst wird verwendet, solange der PowerPointDoktor auf einem Computer mit aktiver Internetverbindung ausgeführt wird. Sollte dies nicht gegeben sein, wird auf die Offline-Übersetzungsfunktion zurückgegriffen. Zu diesem Zweck können im Bereich PowerPoint Babel, der in Abbildung 36 dargestellt ist, Sprachen heruntergeladen werden. Die heruntergeladenen Sprachvorlagen werden in dem Ordner (1) abgelegt. Die Listenansicht (5) zeigt die Sprachmodelle, wobei die Anzeige (2) immer die Übersetzungsmodelle von der Ausgangssprache in die Zielsprache darstellt. Der umgekehrte Weg, d.h. von der Zielsprache in die Ausgangssprache, muss separat über die Liste heruntergeladen werden. Das Icon (3) repräsentiert eine bereits heruntergeladene Sprache, der Button (4) hingegen ermöglicht das Herunterladen einer Sprache.

Abbildung 36:: PowerPoint Babel - Sprachverwaltungsbereich.

Standardeinstellungen für die Übersetzung können im dritten Reiter, dargestellt in Abbildung 37, des PowerPointDoktors vorgenommen werden. Das Setzen einer Standard-Eingabesprache (1) und einer Standard-Ausgabesprache (2) setzt diese für alle in Abbildung 35 angezeigten Quellsprachen (13) und Zielsprachen (15). Mit den Schaltern „Tabellen“ (3), „Grafiken“ (4) und „Bilder“ (5) können Elemente von der Übersetzung ausgeschlossen werden. Sollte „Bilder“ (5) aktiv sein kann über den Radiobutton (6) zwischen den Texterkennungstools `PYTESSERACT` und `KERAS-OCR` unterschieden werden.

Für die PDF-Erstellungsfunktion **(11)** aus Abbildung 35 kann mit (7) eine Zeichenkette angegeben werden. Alle in der PowerPoint-Präsentation gefundenen Textelemente werden bei der PDF-Dokument-Erstellung auf diese Zeichenkette geprüft. Das Erkennungsschema ist folgendes: Wird beispielsweise die Zeichenfolge „===“ angegeben, so muss der überprüfte Text die Form „===Titel===“ haben, damit er von der Funktion richtig erkannt wird. Bei der Angabe der Form (8) muss lediglich die gewählte Form auf der entsprechenden Folie platziert werden. Der PowerPointDoktor unterstützt die Formen Rechteck, Kreuz, Herz sowie Diamant.

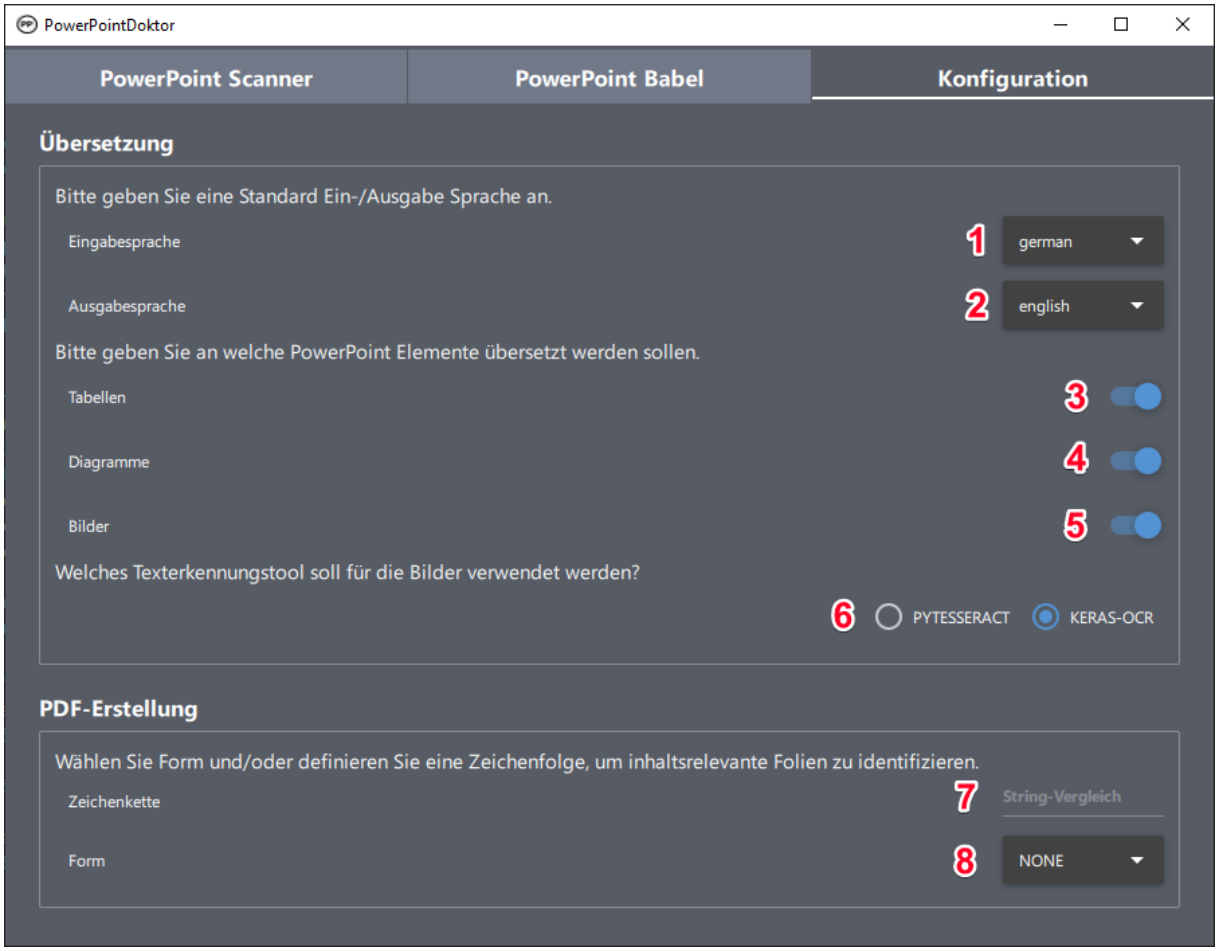


Abbildung 37:: Konfiguration - Einstellungsbereich.

A.2.1: Generierung von Test Präsentationen

Der PowerPointDoktor wird mit einer Klasse zur Erzeugung von PowerPoint-Präsentationen ausgeliefert. Um diese Funktion zu nutzen, muss jedoch der in Abbildung 38 gezeigte Programmcode angepasst werden. Die Zeile 28 zeigt die Variable `generation` die auf `True` gesetzt werden muss. In Zeile 58 können die Parameter nach belieben gesetzt werden um Präsentationen in englischer Sprache zu generieren. Mit `presentations_count` wird die Anzahl der generierten Präsentationen, mit `slides_count` die Anzahl der Folien Seiten pro Präsentation und mit `picture_count` die Anzahl der Bilder pro Präsentation festgelegt. Wichtig: Generierung von 10 Bildern auf 20 Folien bedeutet, dass auf den ersten 10 Folien die Bilder platziert werden bis alle aufgebraucht sind. Für diese Klasse wurde die Bibliothek `TextRecognitionDataGenerator` verwendet.
Mehr dazu kann unter der offiziellen Dokumentation nachgelesen werden:
`https://textrecognitiondatagenerator.readthedocs.io/en/latest/index.html`.

```
26  if __name__ == "__main__":
27
28      generation = False
29
30      if not generation: ···
57      else:
58          pptxGenerator = PptxGenerator(presentations_count = 3, slides_count = 50, picture_count = 50)
59          pptxGenerator.createPresentations()
```

Abbildung 38:: Anpassen der `main.py`, um den PptxGenerator zu nutzen.

A.3: Erzeugen der ausführbaren Datei

Die ausführbare Datei wird mit Hilfe der `PyInstaller` Bibliothek erstellt. Hierfür wird eine Konfigurationsdatei benötigt die in Listing A.2 gezeigt ist. Die Generierung der ausführbaren Datei erfolgt innerhalb des `/src` Ordners über den Konsolen-Befehl *pyinstaller main.spec*. Im Anschluss befindet sich im Unterordner `/dist` die ausführbare Datei `PowerPointDoktor.exe`.

Weitere Informationen zu `PyInstaller` gibt es hier: `https://pyinstaller.org/en/stable/`.

```
# -*- mode: python ; coding: utf-8 -*-
import sys # added line
from os import path # added line
from PyInstaller.utils.hooks import collect_submodules
site_packages = next(p for p in sys.path if 'site-packages' in p) # added line

hiddenimports = []
hiddenimports += collect_submodules('keras')

a = Analysis(
    ['main.py'],
    pathex=[], binaries=[],
    datas=[(path.join(site_packages,"pptx","templates"), "pptx/templates"),
    ('ui_labels.json', '.'), ('userinterface.qml', '.'), ('icon/*', 'icon/'),
    ('icon/lang_download_icons/*', 'icon/lang_download_icons/'),
    ('C:\\Program Files\\Tesseract-OCR', 'Tesseract-OCR'),
    ('lang_models/*', 'lang_models/'),],
    hiddenimports=hiddenimports,
    hookspath=[],
    hooksconfig={},
    runtime_hooks=[],
    excludes=[],
    win_no_prefer_redirects=False,
    win_private_assemblies=False,
    cipher=None,
    noarchive=False,
)
pyz = PYZ(a.pure, a.zipped_data, cipher=None)

exe = EXE(
    pyz, a.scripts, a.binaries, a.zipfiles, a.datas, [],
    name='PowerPointDoktor',
    debug=True,
    bootloader_ignore_signals=False,
    strip=False,
    upx=True,
    upx_exclude=[],
    runtime_tmpdir=None,
    console=True,
    disable_windowed_traceback=False,
    argv_emulation=False,
    target_arch=None,
    codesign_identity=None,
    entitlements_file=None,
)
```

Listing A.2: Inhalt der Konfigurationsdatei `main.spec`.