



**A G H**

**AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W  
KRAKOWIE**

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,  
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

**Praca dyplomowa**

*Opracowanie systemu do analizy informacji na etykietach paczek produktów*

*Development of a system for analyzing information on package labels of products*

Autor:

*Michał Motyl*

Kierunek studiów:

*Automatyka i Robotyka*

Opiekun pracy:

*dr. Katarzyna Grobler-Dębska*

Kraków, 2024







# **Spis treści**

<b>1. Wstęp.....</b>	<b>7</b>
1.1. Cel .....	8
1.2. Motywacja .....	9
1.3. Struktura pracy .....	10
<b>2. Teoria.....</b>	<b>11</b>
2.1. Klasteryzacja oraz klasyfikacja tekstu.....	13
2.1.1. Architektura YOLO .....	15
2.1.2. Sieć YOLOv8.....	17
2.2. Optyczne rozpoznawanie znaków (OCR) .....	20
2.2.1. Tesseract.....	21
2.3. Rozpoznawanie nazwanych jednostek (NER).....	23
2.4. Aktualne zastosowanie metod .....	24
<b>3. Projekt rozwiązania .....</b>	<b>25</b>
3.1. Zbiór danych.....	26
3.1.1. Zbiór danych rzeczywistych .....	27
3.1.2. Generator danych .....	29
3.1.3. Etykietowanie danych .....	31
3.2. Zastosowane narzędzia.....	32
3.2.1. Środowisko wykonawcze.....	33
3.3. Metodyka badania skuteczności rozwiązania.....	33
3.3.1. Klasteryzacja tekstu .....	34
3.3.2. Optyczne rozpoznawanie znaków.....	35
<b>4. Analiza wyników .....</b>	<b>41</b>
4.1. Klasteryzacja tekstu.....	41
4.2. Optyczne rozpoznawanie znaków .....	48

<b>5. Podsumowanie oraz wnioski .....</b>	57
<b>A. Załączniki .....</b>	59
A.1. Ankieta zbierania danych rzeczywistych .....	60

# 1. Wstęp

W toku studiów miałem okazję do zapoznania się z wieloma dziedzinami w których potrzebna jest wiedza oraz umiejętności z dziedziny automatyzacji i robotyki. Jedną z nich, która to bardziej rozbudziła moją ciekawość, była to automatyzacja magazynowa. Zagadnienie jest wielce złożone i wielopoziomowe, dzięki czemu szeroki zakres wiedzy z dziedziny jest wysoce pożądany. Począwszy od zagadnień optymalizacji tras kurierskich, poprzez projektowanie robotów magazynowych na analizie obrazów kamer wewnętrz magazynów kończąc. Problemy, z jakimi mierzy się automatyczny magazyn miałem również możliwość zaobserwować podczas wizyty studyjnej w jednym z największych magazynów automatycznych w Polsce - magazynie firmy Maspex w Tychach. Zainteresowanie wzbudziło we mnie już wtedy zagadnienie odczytywania informacji z etykiet paczek. Proces ten na ogół odbywa się poprzez odczytanie przez magazyniera kodu kreskowego (bądź też kodu QR) z paczki poprzez użycie czytnika. Na tej podstawie uzyskuje się dostęp do szeregu potrzebnych informacji takich jak, przykładowo do kąd dana paczka powinna się udać, jej nadawcę, uwagi dotyczące jej zawartości czy też informacje na temat specjalnego obchodzenia się. Trywialna czynność staje się znaczaco utrudniona kiedy kod dostępny na etykiecie jest uszkodzony. Proste naklejenie przeźroczystej taśmy na fragment kodu może skutkować brakiem odczytu lub też odczytem błędny [1].

Rzadkie przypadki uszkodzenia kodu paczki są najczęściej rozwiązywane poprzez manualne wprowadzenie numeru przesyłki do systemu w celu dostępu do danych. W przypadku, jeżeli numer przesyłki również uległ uszkodzeniu i niemożliwy jest jego odczyt, informacje na etykiecie powinny być wystarczające do jednoznacznego odszukania oraz zidentyfikowania paczki [2]. W opisanej sytuacji informacje z etykiety potrzeba wpisać ręcznie do systemu, co znaczaco spowalnia pracę. Do rozwiązania wynikającego z tego problemu można podejść na dwa sposoby. Pierwszym z nim jest korekcja uszkodzonych kodów kreskowych (bądź kodów QR) algorytmami uczenia maszynowego [3][4]. Proces ten jest złożony i wymaga dużego nakładu czasu oraz pracy, aby rozwiązanie działało prawidłowo. Polega ono na zastosowaniu wyspecjalizowanych algorytmów do odzyskiwania i naprawy uszkodzonych danych na obrazach. Zdjęcie uszkodzonego kodu na etykiecie traktowane jest jako obraz, który musi zostać

odzyskany. Wszystkie te elementy eksponencjalnie zwiększą trudność implementacji oraz złożoność rozwiązania.

Drugim sposobem może być automatyczny odczyt danych bezpośrednio z etykiety poprzez zastosowanie rozwiązań Optycznego Rozpoznawania Znaków (OCR - z angielskiego *Optical Character Recognition*) oraz dalej potencjalnie metod analizy języka naturalnego (NLP - z angielskiego *Natural Language Processing*). Taki sposób najpierw prowadzi do odczytu tekstu z etykiety, a następnie do klasyfikacji danych tekstowych dostępnych w formie pisemnej. W literaturze spotkałem się tylko z jedną inną pracą, która rozwiązuje podobny problem, wykorzystując drugie z opisanych przeze mnie podejścia [5]. Stąd moja decyzja o szczegółowym zbadaniu tematu i opracowaniu rzeczonego rozwiązania samodzielnie, przy jednoczesnej próbie rozwinięcia zastosowanego przez zespół z Uniwersytetu Technicznego w Kaiserlautern. Projekt powstał w roku 2019, od tego czasu powstało bardziej zaawansowane rozwiązanie, w którym poniekąd połączone zostają oba podejścia [1] poprzez wprowadzenie korekcji obrazu przed klasyfikacją jego elementów. Niemniej jednak zdecydowałem się pozostać na systemie bez korekcji obrazu, jak to miało miejsce w wyżej wymienionym systemie.

## 1.1. Cel

Celem pracy magisterskiej jest zaproponowanie systemu do automatycznego odczytu i klasyfikacji informacji znajdujących się na polskich etykietach paczek kurierskich na podstawie obrazu z kamery, bez wykorzystania kodu QR (bądź też kodu kreskowego) znajdującego się na paczce. Z tego względu głównym założeniem systemu będzie bezpośredni brak dostępnych informacji do odczytu z kodu. Informacji, które bezpośrednio potrzebne są do dalszego procedowania przesyłki. System poprzez wykorzystanie algorytmów analizy obrazu będzie klasyfikować, który element etykiety zawiera odpowiednie informacje. Następnie, wykorzystując algorytmy OCR, system będzie odczytywać rzeczone informacje. Dzięki takiemu podejściu można obejść problem związany z potrzebą zastosowania algorytmów rozpoznawania nazwanych jednostek (NER - z angielskiego *Named Entity Recognition*), które w wybranym zagadnieniu nie prezentują obiecujących wyników.

Jako zbiór uczący zostanie zestaw zdjęć wygenerowanych losowych etykiet paczek. Zbiór etykiet rzeczywistych zostanie użyty jako zbiór testowy. Wszystkie etykiety użyte do projektu będą w języku polskim. Etykiety rzeczywiste zostaną zebrane drogą ankiety internetowej z odpowiednim przygotowaniem RODO. Generowane etykiety wypełnione zostaną losowo dobranymi danymi. Utworzono zostanie 6 wariantów etykiet do generacji, od 4 różnych przewoźników na polskim rynku. Każda wygenerowana etykieta zostanie wydrukowana, a następnie sfotografowana na jednym z 3 różnych podkładów zbliżonych do najczęstszych kolorów

paczek. Zarówno do zdjęć etykiet rzeczywistych oraz wygenerowanych przygotowany zostanie odpowiedni opis znajdujący się na nich danych, a także informacji o pozycji rzeczonych danych na obrazie. Dane, które system będzie odczytywał, to kolejno informacje o nadawcy, odbiorcy, oraz uwagach do paczki.

System będzie sprawdzany na trzy sposoby. Po pierwsze sprawdzana będzie zdolność do poprawnego wykrywania informacji na etykiecie poprzez system analizy obrazów. Po drugie sprawdzana będzie zdolność algorytmów OCR do poprawnego odczytu tekstu z obrazu. Następnie sprawdzone zostanie łączne działanie systemu, gdzie algorytmy OCR dostaną do analizy tylko wycinek obrazu zaproponowany odpowiednio przez rozwiążanie lokalizowania elementów na etykiecie. Finalnie sprawdzone zostanie działanie systemu na rzeczywistych etykietach, od których przewoźników system nie widział przykładów podczas uczenia się.

## 1.2. Motywacja

Moją główną motywacją do stworzenia takiego systemu była przede wszystkim ciekawość stworzenia rozwiązania działającego niezależnie od rodzaju etykiety, jaką się użyje do identyfikacji. Trywialnym, ale również pracochłonnym zadaniem wydaje się zakodowanie na stałe wykrycia krańców etykiety, typu, a następnie wartości zależnie od rozpoznanego typu etykiety gdzie można spodziewać się jakiej informacji. System tego typu miałby jedną zasadniczą wadę - nie byłby w stanie przeanalizować etykiet, której nigdy wcześniej nie widział, czy też nie została ona ręcznie wprowadzona do systemu. Każda zmiana wprowadzona przez przewoźnika, czy też wycofanie poprzednich etykiet, lub też zwyczajnie pojawienie się nowego gracza na rynku kurierskim powodowałaby potrzebę edycji systemu.

Z tego powodu szczególnie zainteresowałem się rozwiązaniem, które byłoby na tyle inteligentne, aby analizować etykietę, których wcześniej nigdy nie widziało. Takie podejście sprawia, że wymagana jest znacznie mniejsza ilość czasu przeznaczona na aktualizację i utrzymanie oprogramowania. Skalowalność takiego programu drastycznie wzrasta, przez to także wzrasta zarówno potencjał użytkowy, jak i biznesowy. Drastyczne zmniejszenie przestoi związanych z uszkodzonymi etykietami rozwiązane jednym wspólnym oprogramowaniem niezależnym od przewoźnika jest czymś, co chciałbym kiedyś zobaczyć.

Już w swojej pracy inżynierskiej zajmowałem się zagadnieniami związanymi z analizą etykiet paczek. Przeprowadziłem badania porównawcze działania różnych algorytmów do wykrywania tekstu na etykietach. Porównanie było przeprowadzone zarówno pod względem dokładności detekcji, ale także i pod względem czasu potrzebnego na obliczenia na dwóch różnych platformach sprzętowych - mojej lokalnej maszynie oraz platformie NVidia Json NANO. Ideą było znalezienie balansu pomiędzy złożonością czasową a dokładnością rozwiązań.

Zarówno w pracy magisterskiej jak i w projekcie inżynierskim jako jedno z wykorzystanych rozwiązań zostało wybrane zastosowanie Tesseract. W pracy inżynierskiej [6] analizie zostało poddane działanie silnika Tesseract pod względem nie tylko szybkości obliczeń, ale także i pod względem wpływu zakłóceń na obrazie do finalnego wyniku detekcji. W pracy magisterskiej jego zastosowanie było zgoła inne - bezpośrednio skupiłem się na możliwościach detekcji oraz rozpoznania tekstu, bez wpływu zakłóceń na obrazie. Dodatkowo, zdjęcia mogły zostać poddane obróbce, dostosowanej bezpośrednio do wymagań silnika. Sprawdziłem więc nie jego ogólne możliwości działania względem innych rozwiązań, lecz jego bezpośrednie możliwości maksymalnych osiągów dla jednego wąskiego zastosowania. Jednocześnie w pracy inżynierskiej nie miałem możliwości sprawdzenia zdolności rozpoznania tekstu przez silnik - równie ciekawe zagadnienie, co sama detekcja tekstu. Rozwinięcie moich badań w pracy magisterskiej umożliwiło mi szczegółowe zgłębienie interesującego mnie tematu.

### 1.3. Struktura pracy

W pierwszej kolejności opisuję obecnie stosowane metody wykrywania informacji na obrazie - będą to de facto metody stosowane w wykrywaniu obiektów na obrazie, które zaadoptowane zostały do rozwiązywanego zagadnienia. Następnie przytaczam oraz dokładnie opisuję działanie wybranej przeze mnie metody do wykrywania obiektów na obrazie. Kolejno rozwijam opis teorii o obecnie stosowane metody używane w detekcji tekstu - poświęcam również część rozdziału na dokładny opis wybranej przeze mnie metody. Finalnie w części teoretycznej rozwijam wątek możliwego podejścia NER dla rozwiązywanego problemu. Kolejno przechodzę do opisu samego projektu rozwiązania. Opisuję proces zbierania i przygotowania danych testowych, zastosowane narzędzia w procesie tworzenia kodu, użyte biblioteki w projekcie, oraz hardware użyty do obliczeń. Przedstawiam metodykę weryfikacji działania mojego oprogramowania, aby jednocześnie zanalizować część odpowiedzialną za klasteryzację tekstu, część zajmującą się odczytem tekstu oraz zaprezentować ich współdziałanie. Całość uzupełniona jest opisem procesu fine-tuningu wybranej sieci do wykrywania obiektów na obrazie, oraz wniosków z tejże płynących. Finalnie wyniki poddaje analizie w celach zidentyfikowania mocnych i słabych stron mojego rozwiązania, a także potencjalnych usprawnień projektu.

## 2. Teoria

Problem opracowywany w pracy magisterskiej zasadniczo dzieli się na dwa powiązane ze sobą zagadnienia. Aby zrealizowania zadanie potrzebny jest poprawny i dokładny odczyt tekstu z etykiety, oraz z równie skuteczne przyporządkowanie wykrytemu tekstu odpowiedniej kategorii. W systemie zarówno ważna jest kategoryzacja tekstu jak i jego dokładny odczyt.

Odczyt tekstu na obrazie możemy podzielić na generalne dwa etapy. Wykrywanie pikseli należących do znaków (detekcja tekstu), oraz wykrywanie jakimi znakami jest dany zbiór pikseli (kategoryzacja wykrytych pikseli). Do wykrycia i odczytu tekstu można wykorzystać na ogół dwie metody. Pierwsza z nich to podejścia algorytmiczne, które analizują lokalnie poszczególne piksele i ich otoczenie, aby określić, które z nich mogą należeć do elementów tekstu. Na późniejszych etapach piksele te są grupowane w litery, a litery w słowa. Dopiero po wyznaczeniu kandydatów na litery oraz słowa można przejść do osobnych metod rozpoznających znaczenie danych słów. Drugą z metod detekcji i odczytu tekstu jest równoczesne wykrycie tekstu oraz odczyt znaczenia znaku za pomocą głębokich sieci neuronowych. Nie ma tutaj podziału na etap detekcji oraz odczytu - dzieją się one równocześnie. Podejście do wykrywania tekstu z pomocą sieci neuronowych ma jedną zasadniczą przewagę nad podejściem algorytmicznym: rozwiązuje ono problem wykrycia jak i odczytu tekstu jednocześnie. Faktycznie, podejścia algorytmiczne mają większą dokładność w poprawnym przydzielaniu piksela do znaku. Podejścia algorytmiczne są także znacznie mniej obciążające obliczeniowo, a także mniej kosztowne w przygotowaniu samego rozwiązania (nie trzeba prowadzić procesu uczenia jak dla sieci neuronowych). Niemniej jednak zyski płynące z mniejszej ilości zastosowanych metod do finalnego rozwiązania problemu sprawiają, że sieci neuronowe stały się najbardziej popularną metodą w zagadnieniach rozpoznawania tekstu na obrazach.

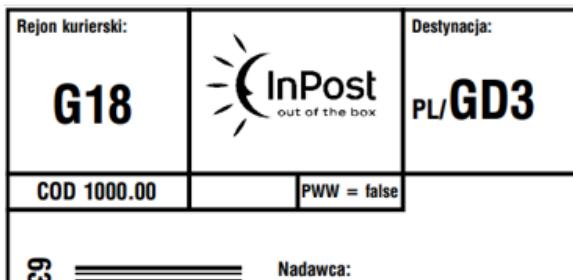
Odczytany tekst sam w sobie dostarcza niewiele informacji w kontekście pozyskiwania danych z etykiet. Bez przyporządkowania odpowiedniej kategorii danemu słowu nie uzyskana zostanie żadna użyteczna informacja - słowo "ul. Nadrzędna" może zarówno być adresem odbioru, nadawcy, czy też odbiorcy (który może być inny niż adres odbioru!). Jak i również może to być

adres firmy kurierskiej (na niektórych etykietach widnieje adres firmy kurierskiej z informacją kontaktową). Dlatego też obowiązkowym jest kategoryzacja wykrytych słów. W podobnych problemach można się spotkać z podejściami nawiązującymi do rozpoznawania nazwanych jednostek (NER - Named Entity Recognition) gdzie mając ciągły tekst kolejno kategoryzujemy każde słowo czy też串 znaków. Innym podejściem, bardziej nakierowanym na problem etykiet, jest zastosowanie algorytmów rozpoznawania obiektów na obrazach w celu kategoryzacji słów przed ich wykryciem za pomocą metod OCR. Rzeczone podejście jest bardziej po-wszechnie ze względu na jego prostotę czy skuteczność działania, ale także równoczesne bardzo szybkie klasteryzowanie tekstu.

## 2.1. Klasteryzacja oraz klasyfikacja tekstu

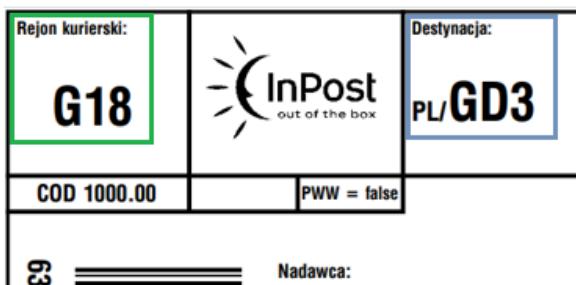
Klasteryzacja czyli grupowanie polega na zbieraniu wcześniej niepołączonych ze sobą obiektów w tzw klastry. Klaster jest kolekcją obiektów które są do siebie "podobne", oraz jednocześnie są "niepodobne" do obiektów należących do innych klastrów [7]. Sama miara podobieństwa czy też niepodobieństwa będzie skrajnie różna zależnie od problemu, który analizujemy. Klasteryzacja tym różni się od klasyfikacji że nie wiemy co reprezentuje dany klaster. Można jedynie powiedzieć, że np. słowa "ul. Nadrzędna numer 3.14 mieszkanie 2.72" należą do wspólnego krastra, ale nie jest się w stanie stwierdzić, czy klaster ten reprezentuje adres zamieszkania, adres odbiorcy, nadawcy czy nawet są to moze uwagi nadawcy umieszczone bezpośrednio na etykiecie. Grupowanie tekstu na obrazie jest o tyle trudne, że zależnie od obrazu sposób grupowania może ulec zmianie. Możliwymi podejściami jest tutaj chociażby zastosowanie algorytmu k-najbliższych sąsiadów do klasteryzacji tekstu [8].

Jakie są korzyści płynące z klasteryzacji tekstu, oraz dlaczego korzystniejsze jest podejście do tego problemu za pomocą analizy obrazów, niż przy zastosowaniu rozpoznawania nazwanych encji (NER)? Na poniższym przykładzie postaram się przybliżyć to zagadnienie. Analizowany będzie fragment etykiety testowej Inpost:



Rys. 2.1. Fragment etykiety testowej Inpost, dostępna na stronie firmy Inpost, z artykułu [9].

Dla przykładu interesującymi polami są rejon kurierski oraz destynacja danej paczki. Dla człowieka jest to trywialne - na etykiecie 2.1 pod napisem "rejon kurierski" znajduje się informacja o rejonie kurierskim, pod napisem "destynacja" znajdują się informacje o destynacji. Jednak jeżeli bezpośrednio odczytany zostanie tekst powyższej etykiety (stosując obecnie powszechnie metody), w wyniku powstanie tekst "linijka po linijce". W najlepszym wypadku będzie to "Rejon kurierski; Destynacja:" jako jedna linijka, oraz "G18, InPost, PL/GD3" jako druga linijka. W rozważaniach pominięta zostaje reszta widniejącej etykiety. Połączenie ze sobą wyrazów z dwóch linijek bez względu do ich pozycję na zdjęciu jest zwyczajnie niemożliwe. Dlatego też klasteryzacja tekstu jeszcze przed jego odczytem jest tak ważna.



**Rys. 2.2.** Fragment etykiety testowej Inpost z ręcznie naniesionymi proponowanymi klastrami. Oryginał bez naniesionych klastrów dostępny w artykule [9].

Teraz jeżeli weźmie się pod uwagę rysunek 2.2 oraz zestawiony on zostanie z wykrytymi słowami, system będzie mógł w prosty sposób stwierdzić, że "Rejon kurierski:" oraz "G18" należą do jakiejś wspólnej grupy, podobnie jak "Destynacja:" oraz "PL/GD3". Jakie są to grupy - albo raczej do jakich kategorii te słowa należą - to już inne zadanie.

Tym innym zadaniem jest klasyfikacja. Każdy wykryty i odczytany fragment tekstu potrzebuje mieć przypisaną mu kategorię - czy to adres odbiorcy, adres nadawcy, uwagi nadawcy, gabyrt lub inne z dostępnych informacji na etykiecie. Bez informacji o tym co dany tekst opisuje, system nie będzie użyteczny w najmniejszym stopniu. Ogólny problem klasyfikacji jest bardzo szeroko znany. Powstało wiele różnych rozwiązań tego zagadnienia - zależnie od specyfikacji można zastosować drzewa decyzyjne, algorytmy SVM (Support Vector Machines), regresje logistyczną czy też sieci neuronowe. W problemie analizy tekstu z obrazu - czyli w problemie z którym będzie zmagać się proponowany system - są dwa zasadnicze podejścia. Pierwszym jest kolejność kiedy najpierw odczytywany jest tekst, a następnie odczytany tekst poddajemy klasyfikacji np algorytmami NER. Drugim jest wpierw klasyfikacja tekstu na podstawie samego obrazu, poprzez nałożenie na obraz ramek (z angielskiego *bounding-box*) a następnie odczytanie tekstu wewnętrz nich.

Podejście zaprezentowane przez zespół z Pekinu [8] jest o tyle nowatorskie, że działało wysoce poprawnie niezależnie od zaprezentowanego obrazu. Jako że moje rozwiązanie będzie miało zawężony zakres działania (etykiety paczek kurierskich), można wykorzystać mnóstwo uniwersalną metodę. Na przykład taką, która pozwalałaby na równoczesną klasyfikację oraz grupowanie tekstu na etykiecie. Jednym z takich podejść jest wykorzystanie bounding-boxów do jednoczesnego grupowania oraz klasyfikacji tekstu [5]. W tym celu może zostać użyta sieć neuronowa w architekturze przystosowanej do problemu detekcji obiektów na obrazie, którą przystosuję do badanego zagadnienia poprzez proces fine-tuningu.

### 2.1.1. Architektura YOLO

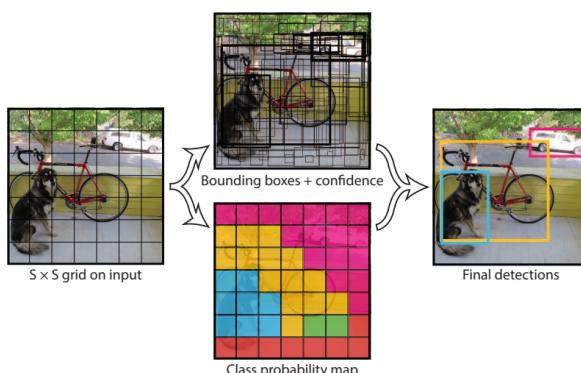
Sieć YOLO [10] - You Only Look Once, czyli z angielskiego spójrza tylko raz, została stworzona i zaprezentowana w roku 2016 jako pierwsze tego rodzaju podejście end-to-end w detekcji obiektów na obrazie. Nazwa wywodzi się z prostej zasady działania sieci - wystarczy tylko jedno przejście obrazu przez sieć aby uzyskać pełną detekcję. Było to o tyle nowe, że wcześniejsze podejścia wymagały wielu przejść przez sieć oraz przez obraz w celu osiągnięcia detekcji. Najbardziej prymitywne podejście z zakresu ruchomego okna wymagało wielu dziesiątek jak nie setek przejść przez sieć w zależności od wielkości i złożoności obrazu (podejście stosowane wcześniej np. w analizie zdjęć fotografii lotniczej [11]). Metody bardziej zaawansowane jak na tamten czas dzieliły zadanie detekcji obrazu na dwa etapy. Pierwszym z nich było wyznaczenie w których obszarach na obrazie może zostać wykryty obiekt (tzw. propozycje regionów, ang. *regions proposals*), gdzie dopiero w drugim etapie wyznaczone obszary były przeliczane przez sieć konwolucyjną w celu wyznaczenia potencjalnego obiektu. Ze względu na połączenie kroku wykrycia regionów oraz kroku użycia sieci konwolucyjnej takie podejście miało nazwę R-CNN [12](ang. *Region-based Convolutional Neural Networks*). Algorytm YOLO nie powstał bez wcześniejszych podstaw. Architektury takie jak SSD (ang. *Single-Shot Multibox Detector*) która skupiała się na wyeliminowaniu kroku propozycji regionów, EfficientDet [13] skupiającą się na podniesieniu efektywności architektury czy też opublikowanej wcześniej RetinaNet [14] skupiającej się na efektach różnego rozkładu klas na obrazach względem pierwszego-drugiego planu, wszystkie mocno wpłynęły na architekturę sieci YOLO, aby wymienić tylko kilka z nich [15].

Architektura sieci YOLO - a dokładniej YOLOv1, Ponieważ odniesienie dotyczy tutaj podstawowej wersji tej architektury, która następnie była wielokrotnie rozwijana - polega na wykryciu wszystkich ramek obiektów znajdujących się na obrazie naraz. Pierwszym krokiem jest podzielenie obrazu na siatkę o wymiarze  $S \times S$ . Jeżeli środek jakiegoś obiektu wypada w danej komórce siatki, ta komórka będzie odpowiedzialna za wykrycie danego obiektu. W pierwszej wersji YOLO założono, że w jednej komórce mogły się znajdować maksymalnie 2 obiekty ( $B = 2$ ). Każda komórka przewiduje  $B$  ramek oraz miar pewności powiązanych z nimi. Miara pewności mówi nam o tym jak pewny jest model, że dana ramka zawiera dany obiekt, oraz jak pewny jest dokładności wyznaczonej ramki. Formalnie za liczenie pewności odpowiada wzór  $P_c = Pr(\text{Object}) * IOU_{pred}^{truth}$  gdzie  $Pr(\text{Object})$  to prawdopodobieństwo że obiekt jest w danej komórce, a  $IOU_{pred}^{truth}$  to miara przecięcia nad sumą (ang. *Intersection over Union*, stąd skrót IoU) pomiędzy przewidywaną ramką a ramką rzeczywistą. Naturalnie, jeżeli obiektu nie ma w danej komórce, pewność powinna wynosić 0.

Każda przewidziana ramka składa się z 5 parametrów:  $x, y, w, h$  oraz miary pewności istnienia obiektu  $P_c$ . Para parametrów  $x, y$  określa nam centrum ramki względem granic komórki w której przewidujemy. Parametry  $w, h$  określają nam odpowiednio szerokość jak i wysokość ramki, względem całego obrazu.

Dodatkowo, w każdej komórce obliczane jest  $C$  prawdopodobieństw warunkowych,  $Pr(Class_i|Object)$ . Te prawdopodobieństwa są uwarunkowane tym, czy dana komórka zawiera obiekt. Niezależnie od tego jakie jest dobrane  $B$  przewidywany jest tylko jeden zestaw prawdopodobieństw.

Jako ostatni krok wartości obliczanych prawdopodobieństw są przemnażane z miarą pewności dla każdej ramki, co daje miary pewności dla każdej z klas dla każdej ramki. Ta finalnie wyliczona miara informuje jednocześnie o tym, jaka jest pewność, że obiekt danej klasy rzeczywiście znajduje się w ramce oraz na ile dobrze dana ramka obejmuje rzeczony obiekt. Działanie jest to zobrazowane na przykładzie 2.3.

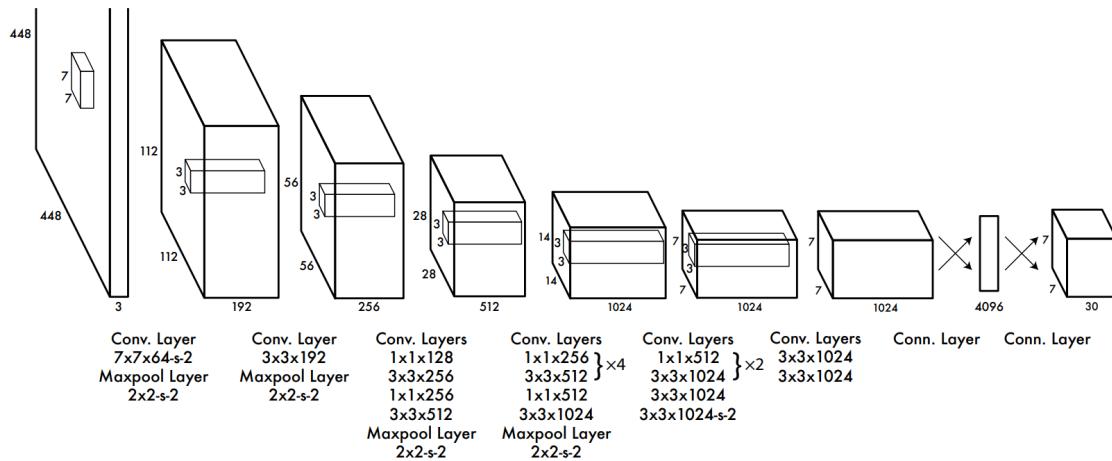


**Rys. 2.3.** Ilustracja działania modelu YOLO. Najpierw następuje podzielenie obrazu na siatkę  $S \times S$ , na każdą komórkę z siatki przewidujemy  $B$  ramek, miare pewności dla rzeczonych ramek, oraz  $C$  prawdopodobieństw klasowych. Przewidywania systemu zapisywane są jako tensor o wymiarach  $S \times S \times (B * 5 + C)$ . Do ewaluacji sieci YOLOv1 użyto  $S = 7, B = 2$  na zbiorze danych PASCAL VOC [16] gdzie dostępnych jest 20 różnych klas obiektów, dlatego też  $C = 20$  co daje nam wynikowy tensor  $7 \times 7 \times 30$ . Ilustracja z artykułu [10].

Jako ostatni krok w każdej ramce wybierane są akceptowane ramki poprzez algorytm NMS [17] (ang. *Non-Max Supression*) oraz progowanie. Naturalnie wartości pewności jakie określone zostaną poprzez progowanie mogą być zmieniane ręcznie w zależności od tego na jakiej mierze działania systemu będzie zależeć w danym przypadku. Analizę wartości progów w detekcji obiektów można przeprowadzić chociażby za pomocą wykresów pewności vs precyzji lub pewności vs recall. Jeżeli jakaś ramka osiągnęła wystarczającą pewność dla istnienia obiektu oraz

najwyższe prawdopodobieństwo z wyliczonych  $C$  prawdopodobieństw również osiągnęło zadanego próg, dana ramka będzie uwzględniona jako odpowiedź na problem detekcji obiektów na obrazie.

Na rysunku 2.3 można zobaczyć intuicyjne uproszczone działanie sieci. Na rysunku 2.3 nie ma jednak samej architektury sieci - czyli defacto sposobu w jaki sieć uzyskuje wynik. Autorzy architektury YOLO inspirowali się tutaj chociażby architekturą sieci GoogLeNet [18] przygotowaną do konkursu w detekcji obiektów na obrazie ILSVRC2014 [19] na którym to sieć GoogLeNet zajęła ex quo pierwsze miejsce. Sieć YOLO składa się z 24 warstw konwolucyjnych oraz 2 warstw w pełni połączonych. Wejściowa rozdzielcość obrazu do sieci to  $448 \times 448$  pikseli. Szczegółowy opis architektury można zobaczyć na poniższym diagramie.



Rys. 2.4. Diagram przedstawiający szczegółową architekturę sieci YOLOv1.

Diagram z artykułu [10].

Należy ponownie podkreślić, że od momentu powstania pierwszej wersji sieci YOLO minęło już wiele czasu - 8 lat w skali rozwoju algorytmów uczenia maszynowego i ogólnego data science to prawie jak zupełnie inna epoka. Do tego czasu powstało wiele iteracji i udoskonaleń oryginalnej sieci, poprzez wersje YOLOv2 aż do YOLOv8 powstałej w roku 2023, o wersjach pobocznych takich jak YOLOR [20] czy YOLOX [21] nie wspominając.

## 2.1.2. Sieć YOLOv8

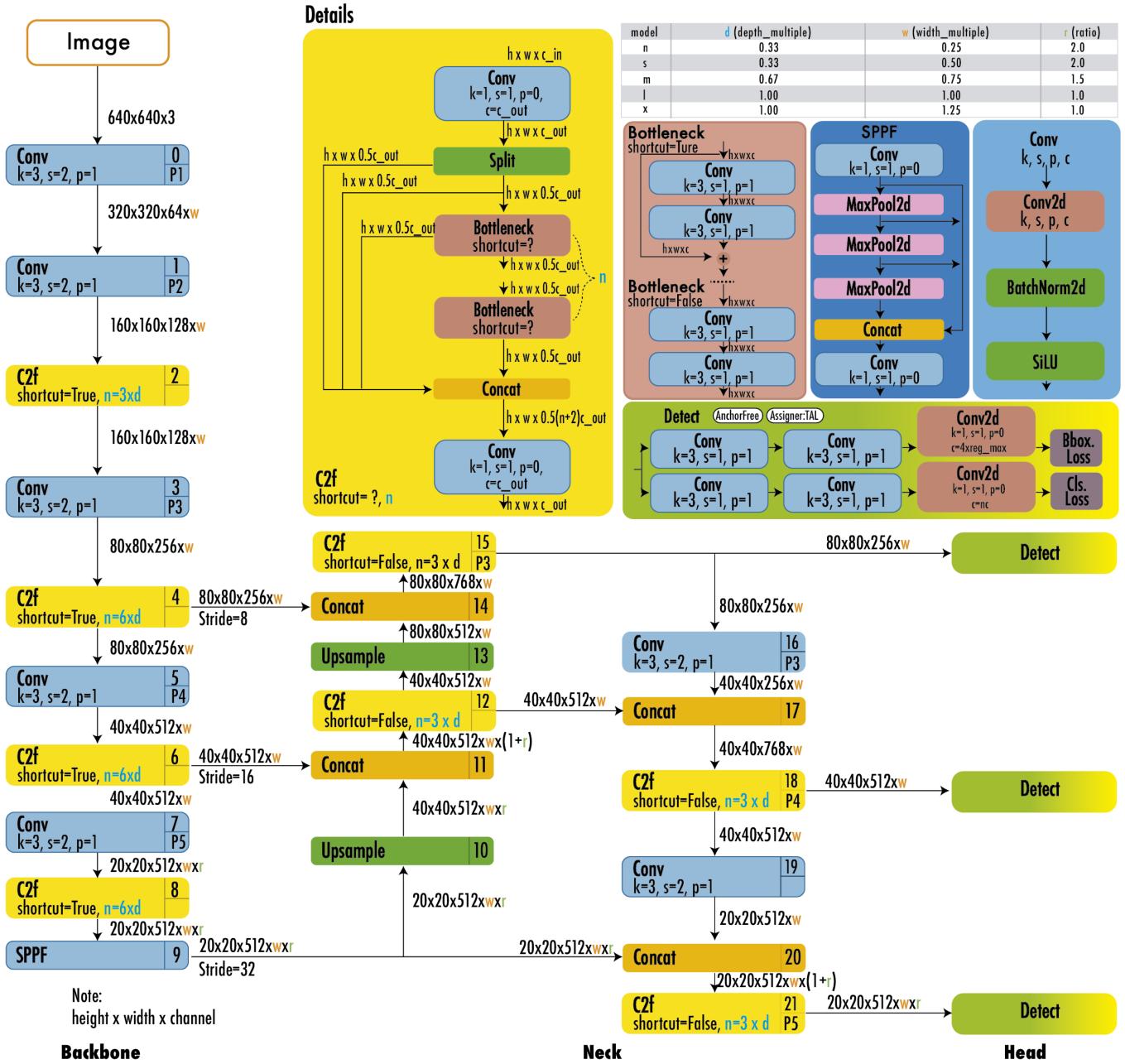
W pracy do zadania detekcji klastrów tekstu na obrazie została wybrana architektura YOLOv8. Jest to jedna z najnowszych iteracji sieci YOLO w chwili obecnej. Sieć YOLOv8 [22] została zaproponowana w roku 2023 przez zespół firmy Ultralytics. Zespół również odpowiedzialny za poprzednie poprzednie iteracje sieci YOLO - YOLOv3 [23] oraz YOLOv5 [24]. Architektura YOLOv8 ma 5 wersji podrzędnych, zależnych od rozmiaru sieci (ilości parametrów):

YOLOv8n (nano), YOLOv8s (mały), YOLOv8m (średni), YOLOv8l (duży) and YOLOv8x (ekstra-duży). Końcówki nazw podrzędnych architektur wynikają z angielskich określeń rozmiaru (*nano, small, medium, large, extra-large*). Sieć YOLOv8 to nie tylko narzędzie do detekcji obiektów, ale także narzędzie do segmentacji obrazów, estymacji pozycji zarówno obiektu jak i osoby na obrazie czy też zadania klasyfikacji. W tej pracy rozwiązanie to zostanie użyte jedynie ze względu na możliwości w detekcji obiektów na obrazie.

Żeby móc opisać architekturę wewnętrzną sieci YOLOv8 potrzeba najpierw cofnąć się do czasów sieci YOLOv3. Wtedy to rozpoczął się trend, w którym większość architektur do detekcji obiektów zaczęła dzielić swoje działanie, na 3 etapy: kręgosłup, szyję oraz głowę sieci (ang. *backbone, neck* oraz *head*). Backbone to zazwyczaj część złożona z wytrenowanej wcześniej sieci konwolucyjnej (CNN), która to wyciąga z obrazu najważniejsze cechy. Następnie szyja jest odpowiedzialny za udoskonalenie i połączenie ze sobą cech wyciągniętych przez backbone. Szyja potrafi łączyć i mieszać ze sobą cechy wyciągnięte przez backbone na różej skali, dzięki czemu informacje przekazywane przez backbone stają się bardziej kompletne. Szyja mogą być dodatkowe warstwy konwolucyjne, mogą to być sieci FPN [25] (ang. *Feature Pyramid Networks*) czy też jakieś inny mechanizmy które pomogą w udoskonaleniu cech wyciągniętych przez poprzednią część sieci. Uproszczony przykład: backbone może powiedzieć sieci, że na obrazie znajdują się koła oraz rama, ale dopiero szyja połączy ze sobą te dwa fakty - sieć będzie wtedy mogła powiedzieć, że jest to jeden obiekt. Kiedy cechy wyciągnięte przez backbone zostały już dopracowane przez szyję, pozostaje nazwać wykryte obiekty - tutaj zaczyna działać głowa. Głowa sieci wykorzystuje ulepszone cechy szyi aby dokonać predykcji.

Na rysunku 2.5 można zobaczyć szczegółową architekturę sieci YOLOv8 z podziałem na backbone, szyję oraz głowę sieci. Warto zaznaczyć tutaj że sieć YOLOv8 to sieć w architekturze tzw "decoupled head" stąd 3 różne elementy oznaczone jako "Detect" na diagramie. Jako backbone sieć wykorzystuje zmodyfikowaną architekturę sieci CSPDarknet53 (połączenie backbone sieci YOLOv3, czyli Darknet-53 [23] oraz CSPNet [26]). Znajdują się tam elementy takie jak warstwy konwolucyjne (zainspirowane Darknet-53, który to model miał 53 warstw konwolucyjnych). Moduły częściowych połączeń warstw (od sieci CSPNet, która to badała właśnie wpływ częściowych połączeń na jakość cech) zostały zastąpione przez moduły C2f oryginalne dla sieci YOLOv8. Finalnie jako ostatni element backbone mamy moduł SPPF (ang. *Spacial Pyramid Pooling Fast*) który odpowiedzialny jest za łączenie cech w mapy o zadanych rozmiarach.

Moduły C2f to modyfikacje modułu CSPNet używanego chociażby w sieci YOLOv5. Moduł C2f to dokładniej połączenie między etapowe z częściowym wąskim gardłem, oraz dwoma



Rys. 2.5. Diagram przedstawiający szczegółową architekturę sieci YOLOv8.

Część elementów sieci została uproszczona do modułu (jak np C2f) a szczegółowo opisana jednokrotnie w prawym górnym rogu ilustracji. Każda warstwa konwolucyjna stosuje normalizację próbki oraz używa funkcji aktywacyjnej SiLU. Na diagramie widać również oddzielne gałęzie sieci do różnych zadań predykcji. Diagram z artykułu [15] stworzony na podstawie dokumentacji [22].

warstwami konwolucyjnymi. Dzięki takiemu podejściu można osiągnąć połączenie cech wysokiego poziomu z informacjami kontekstowymi, dzięki czemu zwiększa się dokładność detekcji. Moduł również spełnia założenia strategii ELAN [27], pozwalającej modelowi na bardziej efektywne uczenie się dzięki kontrolowaniu ścieżek gradientowych.

Szyja to połączenie warstw upsample, konkatenacji, modułów C2f oraz warstw konwolucyjnych. Zależnie od tego do jakiego zadania dalej przekazywane są dane - czyli defacto do której głowy - dane przechodzą przez inny zestaw warstw szyi.

Finalnie - głowy sieci. W architekturze sieci YOLOv8 dostępne są 3 oddzielne gałęzie sieci odpowiedzialne za 3 oddzielne zadania. Tego typu architektura to właściwie architektura "decoupled-head". Oddzielna gałąź odpowiada za detekcję, oddzielna za segmentację, a jeszcze oddzielna za klasyfikację. Dzięki takiemu podejściu każda z gałęzi sieci może się skupić tylko i wyłącznie na swoim zadaniu, dzięki czemu skuteczność indywidualnych rozwiązań problemów ulega poprawie, bez pogorszeniu rozwiązań pozostałych problemów.

Dzięki różnorodności zastosowań, prostocie użycia oraz wielu wariantach wydajności sieci (od nano po extra-large) architektura YOLOv8 jest obecnie jedną z najefektywniejszych. Narzędzia dostarczone w formacie ogólnodostępnego kodu stworzonego przez firmę Ultralytics sprawiają, że proces tworzenia i badania rozwiązania jest dostępny dla niemal każdej osoby posiadającej podstawową wiedzę z zakresu programowania. Razem z jej bardzo dobrymi wynikami, wszystkie aspekty sprawiają, że jest to obecnie jedna z najczęściej wybieranych architektur do detekcji obrazu.

## 2.2. Optyczne rozpoznawanie znaków (OCR)

Optyczne rozpoznawanie znaków to dziedzina która zajmuje się zarówno detekcją tekstu na obrazach, jak i jego poprawnym odczytem. Sposoby samego wykrywania tekstu zasadniczo dzielą się na dwie kategorie - na podejście algorytmiczne, lub na podejście związane z zastosowaniem sieci neuronowych. Podejście algorytmiczne zwykle sprowadza się do analizy pojedynczych pikseli i ich otoczenia. Piksele które spełniają sztywno założone warunki są klasyfikowane jako część litery. Takie piksele są następnie grupowane w pojedyncze litery, a litery naturalnie grupowane dalej w słowa. Takie rozwiązanie nie tylko daje znacznie mniej wyników fałszywie pozytywnych (np. wykrycie na paczce litery "i" podczas gdy jest to jedynie część kodu kreskowego), ale zarówno znacząco mniej wyników fałszywie negatywnych. Litery nie są pomijane. Dodatkowo, podejście algorytmiczne jest znacznie mniej obciążone obliczeniowo, co się może przydać jeżeli rozważane jest rozwiązanie na mniej wydajnej platformie sprzętowej. Jednym z takich rozwiązań jest chociażby *Stroke Width Transform* SWT[28] (SWT) w połączeniu z algorytmem MSER [29]. Obydwa te algorytmy badałem w ramach mojej pracy inżynierskiej [6], gdzie sprawdzałem ich działanie w zależności od platform sprzętowych.

Drugim podejściem jest użycie sieci neuronowych. Znacznie bardziej obliczeniowo obciążające system, jak i również o znacznie większym zapotrzebowaniu pamięciowym. Na ogół jest tutaj podobnie jak w podejściu pierwszym niewiele detekcji fałszywie negatywnych. Litery

nie są pomijane. Różnicą jest zwiększona ilość detekcji fałszywie pozytywnych. Pomimo tych wad, to podejście ma jedną bardzo dużą przewagę - pozwala na jednoczesne wykrycie oraz klasyfikację tekstu. Niemniej jednak, rozwiązania do detekcji tekstu przy użyciu sieci neuronowych wcale nie muszą jednocześnie go kategoryzować. Tak na przykład algorytm "EAST detector"[30](*Easy and Accurate Scene Text detector*) jedynie wykrywa tekst, bez jego kategoryzacji. Co więcej, na podstawie mojej pracy inżynierskiej, mogę powiedzieć że wykrywa ten tekst wyjątkowo dobrze.

Najlepsze efekty są osiągnięte się jeżeli obydwa te podejścia zostaną połączone ze sobą. Kiedy najpierw dokonana zostanie detekcja tekstu algorytmem, a następnie dokonana zostanie jej poprawa poprzez sieć neuronową, osiągnięte zostaną znaczco poprawione wyniki. Kiedy to tak dokładne wyniki poddamy klasyfikacji tekstu, szanse na odczytanie linii kodu kreskowego jako litery "i"(i innych tym podobnych błędów) spadają drastycznie ku zeru. Rozwiążany w ten sposób zostaje główny problem związany z detekcjami fałszywie pozytywnymi w sieci neuronowej. Takim podejściem cechuje się silnik Tesseract w jego obecnej wersji [31].

### 2.2.1. Tesseract

Tesseract to silnik do rozpoznawania znaków, który został udostępniony w 2005 roku jako projekt open-source [32]. Od 2006 roku jego rozwój jest wspierany przez firmę Google, bez naruszania zasad otwartości kodu. Ponieważ Tesseract jest wyłącznie silnikiem, aby używać go w projektach w Pythonie, konieczne jest skorzystanie z biblioteki pytesseract. Biblioteka ta obsługuje komunikację z API silnika w celu wykorzystania jego mocy obliczeniowej w projektach informatycznych.

Oryginalnie system Tesseract działał w sposób zbliżony do algorytmów takich jak SWT, wykorzystując rurociąg danych, który krok po kroku analizuje obraz, z kilkoma istotnymi ulepszeniami. Najpierw przeprowadzana jest analiza spójnych składowych, gdzie zapisywane są obrysły tych składowych. W tym etapie obrysły te są również grupowane (poprzez zagnieżdzanie) w *bloby*. Te bloby są następnie łączone w linie tekstu, a wykryte linie są analizowane pod kątem kandydatów na litery — sprawdza się, czy mają one stałą szerokość (ang. *fixed pitch*), czy też litery zajmują "tylko tyle miejsca, ile potrzebują"(ang. *proportional text*). Tekst *fixed pitch* jest bezpośrednio dzielony na indywidualne kandydatury liter, natomiast *proportional text* jest dzielony na kandydatów przy użyciu np. zbiorów rozmytych.

Po wyodrębnieniu słów oraz liter w tych słowach, rozpoczyna się najbardziej interesująca część działania algorytmu - rozpoznawanie liter. Proces rozpoznawania odbywa się w dwóch

etapach: w pierwszym, każde słowo jest analizowane po kolei. Gdy tylko słowo zostanie poprawnie rozpoznane (lub raczej z wystarczającą pewnością), trafia do adaptacyjnego klasyfikatora, który dodatkowo uczy się na podstawie analizowanego tekstu. Dzięki temu każde kolejne słowo jest rozpoznawane bardziej precyzyjnie, z uwzględnieniem wcześniej rozpoznanych słów. Po jednokrotnym przetworzeniu całego tekstu, adaptacyjny klasyfikator przegląda go ponownie, aby zweryfikować swoje przewidywania. Na koniec rozważane są różne hipotezy dotyczące wielkości tekstu, co pozwala np. wykryć tekst napisany mniejszą czcionką.

Kluczowym elementem odpowiedzialnym za wykrywanie tekstu w silniku Tesseract są sieci LSTM [33]. Architektura rozwiązań stosowanych w systemie jest jednak bardziej zaawansowana i znacznie bardziej specyficzna niż standardowa architektura sieci LSTM. Z uwagi na jej dynamiczny charakter, nie została ona dotąd szczegółowo opisana w dostępnych publikacjach naukowych. Co więcej, architektura i sposób działania są regularnie udoskonalane, co sprawia, że wszelkie opisy działania szybko tracą na aktualności. Zmiany regularnie dokonywane są dokumentowane jedynie hasłowy prezentacjami w dokumentacji oprogramowania [31]. Przez szybkie zmiany pojawiające się co jakiś czas podsumowania naukowe działania silnika stają się nie tylko nieaktualne, ale wręcz bez sensu.

Obecnie najnowszą i jednocześnie najlepszą pod względem dokładności wersję Tesseract jest wersja 5 - wprowadzona do ogólnego użytku w roku 2021. Podobnie do poprzedniej wersji 4 najnowsza iteracja projektu również polega na integracji sieci LSTM w działanie silnika. Wersja 5 została przeze mnie użyta w projekcie.

## 2.3. Rozpoznawanie nazwanych jednostek (NER)

*Named Entity Recognition* (NER), czyli rozpoznawanie nazwanych jednostek, to gałąź problemów analizy języka naturalnego. Problemy NER polegają na poprawnej kategoryzacji słowa w języku pisany. Tak na przykład problemem NER będzie wychwycenie oraz sklasyfikowanie danych osobowych z eseji studenckich czy chociażby kategoryzacja słów z artykułów internetowych w celu wykrycia wrażliwych informacji. Problemy NER są jednymi z najbardziej zaawansowanych zagadnień w dziedzinie NLP, a do ich rozwiązania potrzebne są zarówno duże ilości danych jak i odpowiednio zaawansowane modele językowe.

Najbardziej prominentnym modelem jest obecnie architektura DeBERTa [34]. Bazująca na wcześniejszym modelu transformera BERT [35] dodaje tzw. warstwy rozproszonej attencji w celu lepszego rozpoznawania informacji kontekstowych. Dokładniej, w architekturze DeBERTa informacje o położeniu słowa w zdaniu, oraz o innych słowach w nim występujących, przetrzymywane są w osobnych wektorach. Dzięki czemu uzyskuje się znacznie większą skuteczność klasyfikacji słów. Co więcej, architektury BERT (oraz ich pochodne) wytrenowane na tekście w języku angielskim sprawuje się prawie równie tak samo dobrze w innych językach [36]. Takie rozwiązania badaliśmy z zespołem z koła naukowego Industrial Data Science w ramach konkursu *PII Data Detection* organizowanego przez Vanderbilt University w Nashville, Tennessee. Właśnie te badania zainspirowały mnie do testów podejścia NER dla klasyfikacji tekstu na etykietach.

Idea testów była następująca: w pierwszej kolejności następuje odczyt tekstu na etykiecie, a dopiero później jego klasyfikacja siecią DeBERTa. Po przeprowadzeniu wstępnych testów wykazano że takie podejście jest nieodpowiednie dla badanego zagadnienia. Już podczas treningu sieci na samym tekście było jasne, że takie podejście jest nieprawidłowe. Wyniki sieci były zbliżone do losowych wyborów kategorii słowa. Tekst na etykietach nie zawiera żadnych informacji kontekstowych, przez co sieć nie była w stanie wyciągnąć poprawnych informacji. Informacją kontekstową jest samo położenie tekstu na etykiecie, stąd podejście NER nie ma tutaj prawa bytu. Dzięki szybkiemu wyeliminowaniu tego nurtu jako potencjalnego rozwiązania badania mogły się w pełni skupić na innych rozwiązaniach.

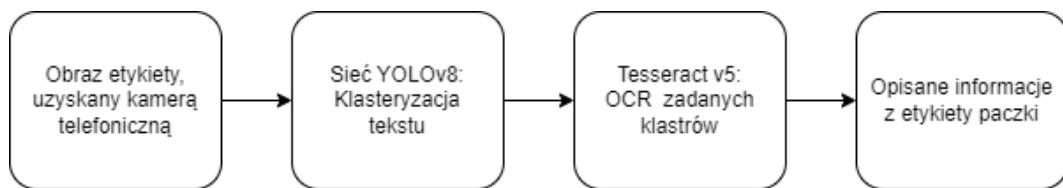
## 2.4. Aktualne zastosowanie metod

Obecnie w detekcji informacji na etykietach widziałem przede wszystkim YOLO w połączaniu z algorytmami OCR. Podejście, które zakłada ekstrakcję informacji bezpośrednio z etykiety, zamiast np. próbować poprawić skan kodu (lub też odczytać numer przesyłki) jest znacznie mniej popularne, przez co zakres stosowanych metod jest znacznie mniejszy. Znacznie częściej rozwiązań skupiają się właśnie na poprawie jakości etykiety, w celu dokonania skanu kodu. Tak na przykład można użyć transformat falkowych w celu poprawy detekcji kodu [1]. Gdy podejście oparte na ekstrakcji informacji bezpośrednio z wydrukowanej etykiety jest brane pod uwagę, sieć YOLO dominuje wśród stosowanych metod. Nie tylko ze względu na prostotę użycia, jasne ramy działania i wysoką skuteczność. Najmocniejszą przesłanką jest prosta przystosowania istniejących już rozwiązań do indywidualnego problemu. Sieć YOLO jest stosowana tak powszechnie, że często stworzenie nowego projektu sprowadza się do nieznaczej adaptacji już istniejącego [37].

Same metody OCR mają szeroki zakres aktualnie stosowanych rozwiązań. Najbardziej popularnym ze względu na swoją prostotę i skuteczność definitelywnie jest silnik Tesseract. W najnowszych wersjach radzi sobie z większością języków które używają alfabetu łacińskiego. Z każdą kolejną wersją dodawane są inne alfabety - jak chociażby kantoński, czy też japońska katakana. Zastosowanie metod takich jak EAST czy też SWT może potencjalnie zwiększyć skuteczność detekcji tekstu poprzez Tesseract, lecz jest mało popularne w zastosowaniach ekstrakcji informacji z zdjęć tekstu. Metody niskopoziomowe właśnie takie jak SWT znajdują zastosowanie gdy potrzeba rozwiązania które nie obciąży obliczeniowo systemu - czyli np. gdy zaistnieje potrzeba przeprowadzenia obliczeń na osobnej platformie obliczeniowej, np. NVidia Nano.

### 3. Projekt rozwiązania

System ekstrakcji informacji z etykiety składa się z 3 modułów. Po wykonaniu zdjęcia obraz z kamery zostaje wprowadzony bezpośrednio do sieci YOLOv8, która wykrywa interesujące nas klastry oraz ich klasy - w obecnej wersji są to nadawca, odbiorca oraz uwagi nadawcy. Sieć YOLOv8 zwraca ramki klastrów, wraz z ich klasami. Na podstawie ramek wycięte zostają fragmenty oryginalnego obrazu, które to przekazane są do kolejnych modułów. Na każdym wycinku zastosowany zostaje silnik Tesseract v5 w celu odczytania znajdującego się na nich tekstu. Tekst odczytany z fragmentów obrazu łączony jest z informacją o klasie klastra z pozyskaną z sieci YOLOv8, aby finalnie uzyskać opisane informacje znajdujące się na etykiecie paczki.



**Rys. 3.1.** Schemat działania systemu. Obraz z kamery zostaje wprowadzony bezpośrednio do sieci YOLOv8. Sieć YOLOv8 zwraca ramki klastrów, wraz z ich klasami. Na podstawie ramek oraz silnika Tesseract v5 odczytujemy znajdujący się na nich tekst. Odczytany tekst wraz z informacją o klasie zwracany jest jako wynik.

Rozwiązanie obecnie skupia się na 3 wcześniej wymienionych kategoriach informacji: adres nadawcy, adres odbiorcy, oraz uwagi nadawcy. Takie ograniczenie zostało nałożone z dwóch powodów: po pierwsze, są to najbardziej podstawowe informacje potrzebne do poprawnego zidentyfikowania paczki kurierskiej w większości przypadków. Po drugie, dodanie trzeciej kategorii jaką są uwagi nadawcy sprawia że projekt poszerza możliwości systemu wcześniejszej przetestowane przez inne zespoły badawcze [5].

### 3.1. Zbiór danych

Specyfika problemu który chciałem rozwiązać wymagała uzyskania zbioru zdjęć etykiet paczek kurierskich, które z definicji zawierają dane osobowe. Dodatkowo, zbiór został ograniczony do etykiet polskich paczek, ze względu na silnie rozwijający się polski rynek dostaw czy też paczkomatów. Ze względu na ochronę danych osobowych, na polskim rynku brak jest ogólnodostępnego zbioru zdjęć etykiet kurierskich. Podczas przeglądu literaturowego zapoznałem się z jedyną pracą, która mówiłaby o udostępnieniu swojego zbioru danych publicznie. Była to publikacja angielskojęzyczna, niemieckiego zespołu - przez co etykiety paczek były w językach angielskim oraz niemieckim. Pomimo obiecanej publikacji zbioru danych, 5 lat po publikacji artykułu zbiór danych nadal pozostaje prywatny. Bezpośrednio kontakt z autorami artykułu nprowadził mnie na potencjalne inne ścieżki w moich badaniach. Finalnie zdecydowałem się na przygotowanie generowanego zbioru danych.

Ze względu na trudny dostęp do danych rzeczywistych wybrano wygenerowanie etykiet kurierskich ręcznie, oraz późniejsze sfotografowanie ich w celu utworzenia zbioru uczącego. Dzięki trzem rodzajom tła jakie są najczęstsze dla paczek które zaobserwowałem (zwykły bezkolorowy karton, biała folia oraz czarna folia) uzyskany został efekt bardzo zbliżony do faktycznych etykiet kurierskich. W generacji wykorzystywane są faktyczne adresy w Polsce, oraz wygenerowane modelem LLM potencjalne uwagi do przesyłki.

Finalnie rzeczony zbiór etykiet wygenerowanych zastosowany został jako zbiór uczącego oraz walidacyjny dla trenowanego modelu, podczas gdy zbiór etykiet rzeczywistych etykiet pozostawiony został jako zbiór testowy do ewaluacji działania całego systemu. Ostatecznie wielkość zbioru uczącego wyniosła 70 etykiet, zbioru walidacyjnego 30 etykiet, a zbioru testowego 21 etykiet.

Po za podziałem na dane rzeczywiste oraz dane generowane, zbiór danych można również podzielić na 3 zasadnicze jego elementy. Na zdjęcie wydrukowanej etykiety, na etykiety zrobionych zdjęć w postaci plików tekstowych z bounding boxami elementów które to będą wykrywane przez sieć YOLO, oraz na pliki csv zawierające informacje o odpowiednio widniejących na etykiecie paczki danych odbiorcy, nadawcy, oraz ewentualnych uwagach nadawcy. Tak przygotowane dane optymalizują dalsze części systemu pod względem do procesu uczenia modelu oraz ewaluacji systemu.

### 3.1.1. Zbiór danych rzeczywistych

Dane rzeczywiste, czyli zdjęcia etykiet paczek wraz z ich słownym opisem zebrane były drogą ankietową. Przy tworzeniu ankiety użyto narzędzia Google Forms. Wybór ten był podyktowany dwoma przesłankami: łatwością późniejszej obróbki danych poprzez środowisko Colab, oraz stosunkowo wysoką dostępność ankiety dla losowego użytkownika. Obydwie te przesłanki wynikają ze specyfikacji w jaki sposób Google Forms przechowuje nadesłane odpowiedzi wraz z plikami obrazu.

Po pierwsze, łatwość obróbki danych wynika z możliwości szybkiego podpięcia swojego osobistego dysku Google do środowiska wykonawczego w jakim pracujemy w Colabie - więcej o tym w rozdziale 3.2. Po drugie, wysoka dostępność dla użytkownika: niezależnie od tego jakie popularne darmowe narzędzie do ankietowania jest rozważane, prawie każde nakłada limity związane z możliwością wgrywania plików jako część odpowiedzi. Tak na przykład ankiety z zestawu Office od Microsoft wymagają aby użytkownik wgrywający plik miał adres mail w tej samej domenie, co domena organizacji pod którą ankieta została stworzona. Ograniczyłyby to zbiór potencjalnych respondentów do osób z aktywnym mailem w domenie "@agh.edu.pl". Google Forms wymaga jedynie, aby użytkownik wypełniający ankietę posiadał adres mailowy w domenie "@gmail.com". Wybór wymagania domeny "@gmail.com" poskutkuje większym potencjalnym zasięgiem ankiety, porównawczo do wymagania domeny "@agh.edu.pl". Link do ankiety wraz z krótkim opisem zagadnienia badawczego opublikowany został zarówno na platformie Facebook jak i LinkedIn, gdzie łącznie uzyskał zasięg ponad 3 tysięcy osób. Ponadto w zbiór danych zaangażowani zostali bliscy krewni oraz znajomi.

Sama ankieta składała się z 3 etapów: opisu słownego przeznaczenia zbieranych danych razem z klauzulą informacyjną wymaganą przez RODO, podstawowych informacji o ankiecie (np. firma przewoźnika) wraz z miejscem na wgranie zdjęcia etykiety, i finalnie opisu zawartości etykiety. Bez podania pozytywnej odpowiedzi na pytanie o zapoznanie się z klauzulą informacyjną niemożliwe jest jej dalsze wypełnianie. Pozytywne zaznaczenie wyżej wymienionego pola równało się bezpośrednio z wyrażeniem zgody na przetwarzanie danych osobowych. Osoby wypełniające ankietę wydawały zgodę na obróbkę ich danych osobowych w zakresie potrzebnym dla pracy magisterskiej, oraz ich użycia jako pojedynczych przykładów działania systemu w samej pracy magisterskiej. Osoby wypełniające ankietę nie dawały zgody na późniejsze opublikowanie ich danych w publicznym zbiorze danych, z tego względu takowy nie mógł powstać. Szczegółowy zakres klauzuli informacyjnej znajduje się w załączniku A.1.

Po zebraniu informacji zwrotnych dodana została też opcja na wrzucenie zdjęcia etykiety bez dodania opisu jej zawartości w celu zwiększenia ilości nadsyłanych odpowiedzi. Każde

pole opisu zawartości w ankiecie było odpowiednio zreferowane, aby nie pozostawić ankietowanym wątpliwości co do sposobu jej wypełnienia. Całość ankiety wygenerowanej do pliku PDF dostępna jest w tym samym załączniku co klauzula informacyjna A.1.

The screenshot shows a survey form section titled "Informacje ogólne". It contains a text input field with placeholder text: "Proszę wrzucić zdjęcie etykiety paczki kurierskiej, którą będzie się opisywać \* (wymaga bycia zalogowania do konta google)". Below it is a file upload button labeled "Upload 1 supported file: image. Max 10 MB." and an "Add File" button. Below this is another section with the question "Czy etykieta **napewno** jest w języku polskim? \*". It includes two radio buttons: "Tak" and "Nie".

Rys. 3.2. Fragment ankiety do zbioru danych rzeczywistych - miejsce do wgrania pliku ze zdjęciem.

The screenshot shows a survey form section titled "Nadawca \*". It contains instructions: "Jeżeli dane nadawcy mają więcej niż jedną linijkę na etykiecie, proszę wpisać całość w odpowiedzi a linijki oddzielić podwójnym slashem (//) . W przypadku braku danych o nadawcy na etykiecie, proszę wpisać znak równości w odpowiedzi (=)." Below is a text input field labeled "Your answer".

Rys. 3.3. Fragment ankiety do zbioru danych rzeczywistych - przykład pola do opisu zawartości etykiety.

Wybór pól do uzupełnienia jako opis zawartości ankiety podyktowany był wcześniej dokonaną analizą dostępnych na polskim rynku usług kurierskich i przewozowych. Po analizie wybrano 13 rodzajów możliwych danych zawartych na etykiecie, niezależnie od przewoźnika. We wstępnie do tego rozdziału wspomnina się o 3 kategoriach, które będą sprawdzane w systemie, lecz przy zbiorze danych rzeczywistych zdecydowano się na poszerzenie tego zakresu, z myślą o potencjalnym rozszerzeniu systemu o kolejne kategorie. W przypadku, jeżeli zaistniał

brak jakiejś danej na etykiecie, użytkownik był proszony o uzupełnienie pola znakiem równości (=).

Finalnie uzyskano zbiór 21 opisanych rzeczywistych etykiet paczek. Zbiór ten posłużył jako sposób na sprawdzenie działania tworzonego systemu jako na danych, których nigdy wcześniej system nie widział. Dodatkowo pozwoliło to na ewentualne sprawdzenie działania systemu w trudniejszych warunkach, jak na przykład pomyte etykiety, zdjęcia słabej jakości, czy też etykiety na których druk zaczął się ścierać.

### 3.1.2. Generator danych

Przy generacji losowych etykiet chciano przede wszystkim, aby były one jak najbardziej zbliżone do rzeczywistych zarówno w wyglądzie jak i w zawartości. Ku temu jako wygląd etykiet zdecydowano się wykorzystać etykiety testowe od firm Inpost (w trzech wariantach), Pocztex, DHL oraz DPD. Jako zawartość etykiet zdecydowano się na wygenerowanie 60 rzeczywistych Polskich adresów. Dzięki temu można było mieć pewność, że dane będą jak najbardziej zbliżone do rzeczywistych. Dane wygenerowane jako krótkie uwagi nadawcy były stworzone za pomocą modelu LLM - ChatGPT4. Wygenerowane uwagi były poddane manualnej weryfikacji, tak, aby uwagi umieszczone na etykietach zawsze miały związek z tematyką nadawanych paczek.

Obecnie dostępne na rynku rozwiązania do generowania etykiet w standardach przewoźników wymagają płatnej subskrypcji, oraz także zastosowania narzuconego przez producentów SDK do obsługi oprogramowania. Z tego względu napisane zostało od podstaw narzędzie do nanoszenia informacji na etykiety. Do tego celu wykorzystany został język programistyczny Python oraz przede wszystkim biblioteka OpenCV, w jej wersji przystosowanej do wybranego języka [38]. Na każdej etykiecie testowej zamalowano na biało wybrane pola, dzięki czemu po zapisie uzyskano 7 różnych wzorów etykiet gotowych do wypełnienia.

Każda z wybranych etykiet celowo różniła się od siebie - czy to rozmiarem, położeniem informacji czy też zawartością (np. Inpost zawiera na etykiecie także informacje o regionie kurierskim, podczas gdy Pocztex zawiera chociażby informacje o wykupionych dodatkowych usługach). Z tego względu dla każdej etykiety powstał osobny preset dla funkcji odpowiedzialnej za uzupełnianie etykiet-blanków o tekst. Jako że dokładne specyfikacje fontu oraz wielkości tekstu na etykietach nie są łatwo dostępne, zdecydowano się na testy metodą prób i błędów w celu uzyskania wyglądu każdego z wypełnianych pól jak najbardziej zbliżonego do rzeczywistej etykiety. Również metodą prób i błędów wyznaczano lokalizację poszczególnych informacji na obrazie (tj. wielkości i lokalizacje ramek, w których tekst będzie wprowadzany poprzez

OpenCV). Finalne, bez dokładnej analizy zawartości tekstu na etykietach, efekty generacji były prawie nie rozróżnialne od etykiet rzeczywistych.



**Rys. 3.4.** Przykład wypełniania etykiety danymi. Od lewej do prawej: etykieta testowa, etykieta-blank oraz etykieta wypełniona losowo dobranymi danymi. Etykiety-blanki były przygotowane jednorazowo przed generacją. Podczas generacji używano wcześniej już przygotowanych blanków. Na przykładzie powyżej etykieta testowa Inpost. W prawym dolnym rogu wypełnionej etykiety widnieje numer wygenerowanej etykiety w celach sprawnej identyfikacji etykiety przez ludzkiego użytkownika po zrobieniu jej zdjęcia.

Wybór danych do wypełnienia danej etykiety był losowy. Wybór informacji adresowych (niezależnie od tego w jaki sposób adres będzie widzieć na etykiecie - nadawca, odbiorca, paczkomat etc.) był zrealizowany poprzez wybór losowego rekordu z listy wcześniej przygotowanych adresów na terenie Polski. Istniała 20% szansa, że dany adres będzie zawierać nazwę firmy. Jeżeli tak się stało, nazwa firmy była losowana ze wcześniej przygotowanej listy. Uwagi nadawcy również były losowane ze wcześniej przygotowanej listy - tak jak wspomniano wcześniej. Dodatkowo, jeżeli na etykiecie pojawiały się pola specjalne (jak np. rejon kurierski, lub też destynacja, czy numer punktu odbioru przesyłki) wybory te również losowano ze wcześniej przygotowanych list. Wszystkie przygotowane listy były zapisane w formacie .xlsx dostępne do oglądu w repozytorium projektu.

Finalnie, w celach późniejszej identyfikacji etykiet do testów, każda etykieta była opisana numerem w prawym dolnym rogu. Dzięki temu po wydrukowaniu etykiet i ponownemu zrobieniu ich zdjęć, można było sprawnie zidentyfikować, które zdjęcie odpowiada której wygenerowanej etykiecie. Dane umieszczone na etykietach zapisywane były zbiorczo do pliku csv.

W celu ujednorodnienia pliku, w przypadku gdy jakaś dana nie była użyta na danej etykiecie (np. informacja o numerze paczkomatu na etykiecie Pocztex) dana kolumna w wierszu w pliku pozostawała pusta. Podobnie jak w przypadku zbioru danych rzeczywistych, zdecydowano się na zapisanie większej ilości informacji, niż było to bezpośrednio potrzebne w danym momencie. Dzięki temu, jeżeli w trakcie testów zdecydowano by się na dodatkową klasę informacji do odczytania z etykiety, natychmiast przygotowane były do tego dane.

Po przygotowaniu plików wygenerowane etykiety zostały wydrukowane, wycięte oraz sfotografowane na 3 różnych tła. Tła zostały wybrane takie, jakie z doświadczenia najczęściej znajduje się na paczkach: zwyczajna tektura, biała folia oraz czarna folia. Zdjęcia zostały wykonane pod nieznacznie różnymi kątami. Na każdym zdjęciu tło jest obrócone względem etykiety w taki sposób, aby niejednorodne fragmenty tła nie wpłynęły w późniejszych etapach na uczenie się modelu. Poprzez zwiększenie wariancji tła można było zapewnić, że model nie nauczy się wykrywać elementów etykiety zależnie od np. zagięcia w kartonie, czy też plamy na białej folii.

### 3.1.3. Etykietowanie danych

Etykietowanie danych podzielone zostało na dwie części - opis zawartości etykiety (do wykrycia poprzez narzędzia OCR) oraz opis klastrów na obrazie (do wykrycia przez sieć YOLO). Opis zawartości był bardzo bezpośredni - etykiety wygenerowane posiadały przygotowany opis ich zawartości, zapisany podczas procesu generacji opisanego wcześniej. Etykiety rzeczywiste w większości także miały przygotowany opis ich zawartości dzięki odpowiedniemu zaprojektowaniu ankiety. W przypadku etykiet rzeczywistych wystarczające było jedynie zweryfikowanie, czy opisy nie zawierają błędów.

Trudniejszym zadaniem było opisanie samych zdjęć, a dokładniej naniesienie na nie odpowiednich ramek. Sieć YOLO posiada ustalony z góry format oznaczenia obiektów do wykrycia na zdjęciach. Tutaj znacząco pomogła popularność rozwiązań *You Only Look Once* - dostępność narzędzi do etykietowania zdjęć jest dzięki temu bardzo szeroka. Zdecydowano się użyć odpowiednio dostosowanego narzędzia CVAT [39] (ang. *Computer Vision Annotation Tool*). Poza dostępnym modelem do etykietowania danych, webowa aplikacja CVAT pozwala także na tworzenie oraz zapisywanie całych projektów data science. Dzięki temu, jeżeli w trakcie tworzenia projektu rozwiązań pojawiłyby się nowe dane, można z łatwością dodać je do zbioru na platformie webowej i opisać wedle już wcześniej zapisanego schematu. Opisane dane można eksportować nie tylko do szablonu YOLO ale także wielu innych, co sprawia, że narzędzie może być używane w różnorakich projektach z zakresu detekcji obiektów na obrazie.

## 3.2. Zastosowane narzędzia

Projekt został wykonany w całości w języku Python, ze względu na wysokie przystosowanie jego bibliotek do zadań związanych z analizą danych oraz uczeniem maszynowym. Wykorzystano zarówno skrypty języka python jak i notebooki Jupyter. Nie jest to obliczeniowo najszyszsze rozwiązanie (choćby biblioteka OpenCV oryginalnie jest przeznaczona do użytku w języku C++, przez co w języku Python działa wolniej) lecz ze względu na stosunkową prostotę rozwoju projektu informatycznego jest to rozwiązanie dzięki któremu najprościej uzyskać informację o działaniu tworzonego projektu.

Przy rozwijaniu systemu oraz w trakcie analizy jego wyników użyto szeregu bibliotek oraz narzędzi. Jako podstawy do obsługi obrazów użyto biblioteki OpenCV, w wersji przystosowanej do języka Python. Zarówno do obróbki, ładowania, zapisu czy też nanoszenia edycji na obrazy - stosowana była biblioteka OpenCV. Do wyświetlania obrazów w trakcie rozwoju rozwiązania oraz testów używana była biblioteka matplotlib [40]. Biblioteka ta nie tylko pomagała sprawdzić czy edycje wprowadzane do obrazów były poprawne, ale także pomagała w analizie danych. Matplotlib zamiennie z biblioteką seaborn [41] służył jako narzędzie do wykresowania wykresów w celu analiz takich jak podział przewoźników w zbiorze rzeczywistym. Biblioteki te są także bezpośrednio wykorzystywane przez bibliotekę Ultralytics do przygotowania oraz analiz tablic kontyngencji (ang. *confusion matrix*) czy też krzywych ROC.

Do generatora danych nie tylko potrzebowano narzędzi do edycji obrazów, ale także do losowego wyboru danych z przygotowanych wcześniej tabel. Ku temu celu wykorzystano przede wszystkim bibliotekę numpy [42] - zarówno do generacji liczb losowych z zakresu, ale także do obsługi list oraz pomocy przy obsłudze obiektów z danymi. Do obsługi tabel, czyli zarówno do generacji jak i do zapisu danych, użyta została biblioteka pandas [43].

Obsługa YOLOv8 była prowadzona poprzez bibliotekę Ultralytics, przygotowaną bezpośrednio do obsługi rzeczywej sieci. Dzięki temu w jednym rozwiążaniu deweloper ma dostęp do narzędzi fine-tuningu sieci, analizy jej wyników, analizy postępów uczenia oraz bezpośredniego wglądu w pojedyncze predykcje. Biblioteka ta również jak wspomniano wcześniej pozwala na automatyczną generację wykresów po skończonym treningu sieci, dzięki czemu deweloper ma bezpośredni wgląd w szczegółowe metryki modelu nad jakim pracuje.

Silnik Tesseract jest obsługiwany poprzez zapytania API, przez co wymagana jest biblioteka do obsługi rzeczywionych zapytań w pythonie. Ku temu została zastosowana biblioteka pyteseract [44]. Do przetwarzania obrazów w celu poprawy działania systemu OCR zastosowana została biblioteka OpenCV. Finalnie do obliczeń miar działania systemu OCR (czyli odpowiednio WER, CER, oraz MER) zastosowana została biblioteka jiwer, bazowana na [45].

### 3.2.1. Środowisko wykonawcze

Pierwotnie środowiska Colab planowa użyć tylko do wytrenowania sieci YOLOv8 aby potem użyć nauczonego modelu w systemie przygotowanym lokalnie. Trening przeprowadzony był na maszynie wirtualnej z dwoma dostępnymi kartami graficznymi, całkowicie 15 GB pamięci GPU RAM, oraz 12 GB pamięci CPU RAM. Ze względu w różnicę w architekturze lokalnej maszyny, a maszyny na której przeprowadzano uczenie, inferencja sieci zosąła w pełni przeprowadzona na środowisku przygotowanym w Colabie.

Na platformie Colab inferencja sieci YOLOv8 przebiegała bezproblemowo, lecz natomiast nie było możliwości instalacji Tesseract w wersji piątej. Na lokalnej maszynie można było dowolnie wybrać wersję Tesseract do instalacji. Stąd podział systemu na dwa etapy: analizę obrazu wykonaną w środowisku Colab, gdzie uzyskano pliki wynikowe, które następnie zapisano narzędziami dostępnymi w bibliotece pickle. W ten sposób zapisane wyniki detekcji na obrazie przesłano do lokalnego systemu. Pliki te były ładowane, i na ich podstawie przeprowadzano detekcję tekstu na obrazie. Również na lokalnej maszynie zostaje wyliczona miara dokładności rozpoznania tekstu. Dzięki takiemu podejściu uzyskano pełną kontrolę zarówno nad efektywnym działaniem YOLOv8 jak i nad dostosowaniem wersji silnika Tesseract do wybranego zastosowania.

W trakcie wykonywania testów na lokalnej maszynie losowe ramki z wykrytym fragmentem obrazu z sieci YOLOv8 oraz naniesione nań detekcje wyrazów Tesseract (wraz z obrazem detekcji po obróbce, przed działaniem nań algorytmem OCR) zostały zapisane w celach analitycznych. Przykłady rzeczowych ramek zawarto w repozytorium projektu, a także zostały one pokazane w późniejszej części pracy.

## 3.3. Metodyka badania skuteczności rozwiązania

Ze względu na podstawowe różnice w działaniu mechanizmu klasteryzacji oraz optycznego rozpoznawania znaków naturalnym jest potrzeba oddzielnej metodologii badania ich skuteczności. W przypadku badania działania sieci YOLOv8 najważniejszym jest sprawdzenie czy przewidziane ramki pokrywają się z ramkami przekazanymi w zbiorach wykorzystanych przy treningu. Naturalnie nie są to jedyne metryki jakich można użyć w trakcie analizy wyników. Dla sieci YOLO np. podstawowe metryki takie jak precyzaja oraz recall dają również dużo informacji. Niemniej jednak są one na tyle znane w dziedzinie, że obszerne omawianie ich w tym dziale wydaje się nie być potrzebne.

W przypadku detekcji tekstu, jako że ostatecznym użytkownikiem systemu w założeniu jest człowiek, potrzeba wiedzieć, czy tekst po detekcji wymaga edycji. Dokładniej, jeżeli tekst został

wykryty nieprawidłowo, ile takowych edycji potrzeba, aby uzyskać tekst tożsamy, z tekstem podlegającym detekcji.

### 3.3.1. Klasteryzacja tekstu

Biblioteka Ultralytics udostępnia bezpośrednio narzędzie do tworzenia wykresów uczenia sieci. Na tych wykresach możemy przeanalizować różne wartości - od licznej funkcji straty "box\_loss" do metryk takich jak precyza, recall oraz odpowiednio mAP50 i mAP50-95. Żeby móc zrozumieć jak działają wymienione metryki w zadaniu detekcji obiektów na obrazie potrzeba wprowadzić definicję IoU - *Intersection over Union*.

IoU to metryka pozwalająca na intuicyjne zrozumienie w jakim stopniu ramki tworzonego modelu pokrywają się z faktycznymi ramkami obiektów na zdjęciach. IoU liczone jest na podstawie poniższego prostego wzoru:

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (3.1)$$

gdzie

- $IoU$  to wartość przecięcia nad sumą
- *Area of Overlap* to pole powierzchni przecięcia hipotezy oraz referencji
- *Area of Union* to pole powierzchni sumy zbiorów hipotezy oraz referencji

Z angielskiego *area of overlap* to pole powierzchni wspólne dla ramki przewidzianej, oraz ramki faktycznej. *Area of union* to wspólne pole powierzchni dla obu ramek. Im wartość IoU jest bliższa wartości 1, oznacza to, że tym dokładniej model przewiduje pozycje ramek. Wartość równa 1 oznacza tożsamość ramki predykcji z ramką faktyczną. Na podstawie wartości IoU możemy definiować kiedy dany obiekt jest uznany za wykryty.

Kolejnymi metrykami godnymi omówienia są mAP50 oraz mAP50-95. Obydwie metryki odnoszą się do średniej wartości precyzji dla całego modelu. Wartość mAP50 liczona jest z ustalonym poziomem  $IoU$  równym 0.5 - jeżeli wartość  $IoU$  przekroczy 0.5, obiekt uznawany jest za wykryty. W przypadku mAP50-95 koncept ten jest rozszerzany. Uśredniona zostaje wartość precyzji detekcji nie tylko niezależnie od klasy obiektu, który wykrywamy, ale także niezależnie od poziomu IoU (dokładniej od 0.5 do 0.95, stąd cyfry w nazwie metryki) dobranego dla testów. Metryka mAP50-95 dokonuje kalkulacji średniej precyzji dla różnych poziomów IoU, przez co uzyskiwana jest znacznie bardziej przekrojowa informacja na temat działania modelu.

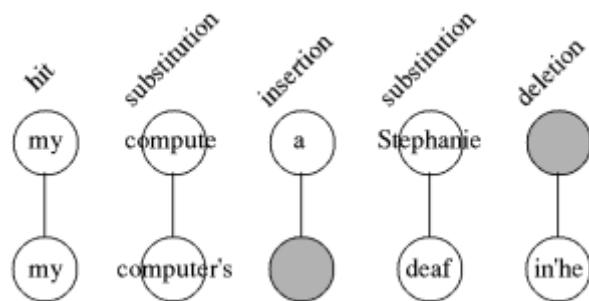
Metryki uśredniające takie jak wyżej wymienione mają problemy w przypadku, kiedy w zbiorach uczących zachodzi duża dysproporcja klas. Nie uzyskuje się przez to informacji o

tym, która klasa odpowiedzialna jest za zaniżanie wyników. Ten problem można rozwiązać na dwa sposoby - wykreślić wykresy z podziałem na klasy, lub też zadbać o balans klas w zbiorze. W projekcie wybrano podejście z balansem klas w zbiorze. Jako że zbiór generowano, problem ten został rozwiązyany już na etapie generacji.

Gdy już rozwiązany zostanie wcześniej wymieniony problem, można na podstawie wartości metryk bardzo szeroko i wyczerpująco analizować działanie modelu. Jeżeli mAP50 oraz mAP50-95 są na zbliżonym do siebie poziomie można stwierdzić, że model dobrze przewiduje ramki. Im bardziej rozbieżne są wyżej wymienione wartości, tym bardziej model będzie mieć problemy z poprawnym przewidzeniem ramek obiektu. Porównanie tych dwóch metryk dla tego też jest najważniejszą informacją o działaniu modelu YOLOv8. Ze względu na wygodę użytkowania systemu preferowane jest, żeby IoU było jak największe, nawet jeżeli miałyby to być kosztem precyzji. Ważniejsze w koncepcie działania projektu było podanie dokładniej detekcji, niż jakiekolwiek detekcji. Dzięki temu, kolejny fragment systemu, czyli rozpoznawanie znaków, będzie w stanie poprawnie zaklasyfikować tekst jako dotyczący odbiorcy, nadawcy czy uwag. Im dokładniej wyznaczone zostaną ramki, tym mniej przypadkowego tekstu wykryje moduł OCR.

### 3.3.2. Optyczne rozpoznawanie znaków

Skuteczność wykrytego tekstu na obrazie obliczana się tak samo, jak chociażby skuteczność wykrytego tekstu z czyjejś mowy czy też nagrania wideo. Najważniejszą informacją jest miara, ile pojedynczych edycji należy dokonać, aby z hipotezy systemu (tekstu uzyskanego przez jego działanie) otrzymać tekst faktyczny czyli tzw. referencyjny. Do badanego problemu wybrane zostały miary WER, CER oraz MER [45]. Nie są to bezpośrednie wartości ilości rzecznego edycji, lecz miary bezpośrednio z nich wynikające. Na ich podstawie możliwym jest wnioskowanie o użyteczności detekcji np. w zależności od późniejszego jej przeznaczenia.



Rys. 3.5. Przykład jak interpretować czym jest  $H$ ,  $S$ ,  $D$  oraz  $I$ . Z angielskiego odpowiednio *hit*, *substitution*, *deletion*, *insertion*. Przykład z [45]

WER - czyli *Word Error Rate* to podstawowa miara, jaka jest używana przy kalkulacjach związanych z detekcją słów. Jest to proporcja błędów w detekcji, do słów które procedujemy. Kiedy  $H, S, D$  oraz  $I$  to odpowiednio ilość poprawnych trafień, zastąpień, usunięć oraz wstawień potrzebnych do uzyskania referencyjnego tekstu z uzyskanej przez system hipotezy (przykład 3.5). Wtedy:

$$\text{WER} = \frac{S + D + I}{H + S + D} \quad (3.2)$$

gdzie

- WER to wartość *word error rate*
- $H$  to ilość poprawnych trafień słów
- $S$  to ilość zastąpień słów
- $D$  to ilość usunięć słów
- $I$  to ilość wstawień słów

Dodatkowo jeżeli zdefiniujemy  $N_1, N_2$  oraz  $N$  jako odpowiednio ilość słów na wejściu (referencja), na wyjściu (hipoteza systemu) oraz ilość par słów in/out (przykład 3.5), to można wtedy zdefiniować maksymalną możliwą wartość WER dla danej detekcji. W przypadku detekcji pojedynczych słów (tzw. IWR -*Isolated Word Recognition* będzie to:

$$\text{Max WER} = \frac{\max(N_1, N_2)}{N} \quad (3.3)$$

gdzie

- Max WER to maksymalna możliwa wartość WER dla danej detekcji IWR
- $N_1$  to ilość słów referencji
- $N_2$  to ilość słów hipotezy
- $N$  to ilość par słów 'in-out'

W przypadku detekcji całych wyrażeń (tzw. CSR - *Connected Speech Recognition*) będzie to wtedy:

$$\text{Max WER} = \frac{\max(N_1, N_2)}{N_1} \quad (3.4)$$

gdzie

- Max WER to maksymalna możliwa wartość WER dla danej detekcji CSR
- $N_1$  to ilość słów referencji
- $N_2$  to ilość słów hipotezy

Przypadek działania systemu to problem CSR. Naturalnym celem jest, aby wartość WER była jak najmniejsza - im mniejsza, tym mniej błędów do poprawy dla ludzkiego użytkownika.

Wartość WER jest najlepszą wartością do zastosowań kiedy wynik detekcji może zostać poprawiony przez człowieka. Nie jest to jednak jedyna przydatna metryka. Metryka MER (*Match Error Rate*) jest bardzo zbliżona do WER, lecz daje nam delikatnie inną informacje. Daje nam prawdopodobieństwo danej detekcji bycia poprawną. Miara MER i WER bardzo często będą dawać tożsame ze sobą wyniki, lecz istnieją przypadki gdzie zaobserwowanie w nich różnic może dać dużo informacji na temat możliwych poprawek w systemie. Wartość MER daje bezpośrednią информацию o prawdopodobieństwie, czy dane słowo jest błędnie wykryte. Stąd wartość MER również powinna podlegać minimalizacji w systemie. Jako że wartość MER to wartość prawdopodobieństwa, maksymalną możliwą wartością dla miary MER jest 1.

$$\text{MER} = \frac{S + D + I}{H + S + D + I} \quad (3.5)$$

gdzie

- MER to wartość *match error rate*
- $H$  to ilość poprawnych trafień słów
- $S$  to ilość zastąpień słów
- $D$  to ilość usunięć słów
- $I$  to ilość wstawień słów

Finalnie przydatnym jest spojrzeć na pojedyncze znaki. Powyższe miary patrzyły na całe słowa, lecz jeżeli będziemy chcieli zautomatyzować nasz system, informacja o tym jak blisko dane słowo jest temu co przewidujemy jest równie ważna. Stąd wybór miary CER (*Character*

*Error Rate*). W tym wypadku wartości  $H, S, D$  oraz  $I$  odnoszą się nie do całych słów, lecz do pojedynczych znaków.  $N$  jest liczbą wszystkich znaków do detekcji. Wtedy:

$$\text{CER} = \frac{S + D + I}{N} \quad (3.6)$$

gdzie

- CER to wartość *character error rate*
- $H$  to ilość poprawnych trafień znaków
- $S$  to ilość zastąpień znaków
- $D$  to ilość usunięć znaków
- $I$  to ilość wstawień znaków

Ze względu na odmienny charakter miary CER wartość maksymalna liczona jest odmiennie od maksymalnej wartości WER, niemniej jednak podobieństwa są uderzające. Dla CER wartości  $N_1, N_2$  to nie ilości słów, tylko ilości wykrytych znaków. Wtedy:

$$\text{Max CER} = \max(N_1, N_2)/N_1 \quad (3.7)$$

gdzie

- Max CER to maksymalna możliwa wartość CER dla danej detekcji
- $N_1$  to ilość znaków referencji
- $N_2$  to ilość znaków hipotezy

Co należy bezpośrednio zaznaczyć, to że dla miar CER oraz WER porównanie ze sobą miar większych od 1 może być mylące. Dla przykładu: system A oraz system B. W momencie w którym wartości CER czy też WER są większe lub równe jeden, jest to tożsame z sytuacją kiedy to ilość wstawień potrzebnych do uzyskania tekstu referencyjnego na podstawie pustego tekstu jest mniejsza, niż ilość wszystkich edycji potrzebnych do uzyskania tekstu referencyjnego z tekstu hipotezy. Jeżeli system A miałby średnią wartość WER 1.1, a system B - 1.7, nie można jednoznacznie stwierdzić, który system działa gorzej. Można jedynie stwierdzić, że zarówno system A jak i system B nie dostarcza żadnej użytecznej informacji, jako że obie wartości są większe od 1.

Wszystkie wymienione wyżej miary powinny być minimalizowane. WER przekazuje najwięcej przydatnych informacji, jeżeli system będzie używany przez końcowego użytkownika

ludzkiego (im mniejszy WER tym mniej pracy dla użytkownika). MER zawier informacje o tym, jak prawdopodobne jest niepoprawne wykrycie słowa. CER daje informacje ile znaków może być błędnych. Połączenie MER oraz CER może być wykorzystane jeżeli chcemy wnioskować na temat użyteczności systemu, jeżeli na końcu to nie ludzki użytkownik będzie analizować wyniki, lecz np. algorytm szukający podobnych adresów w bazie danych.



## **4. Analiza wyników**

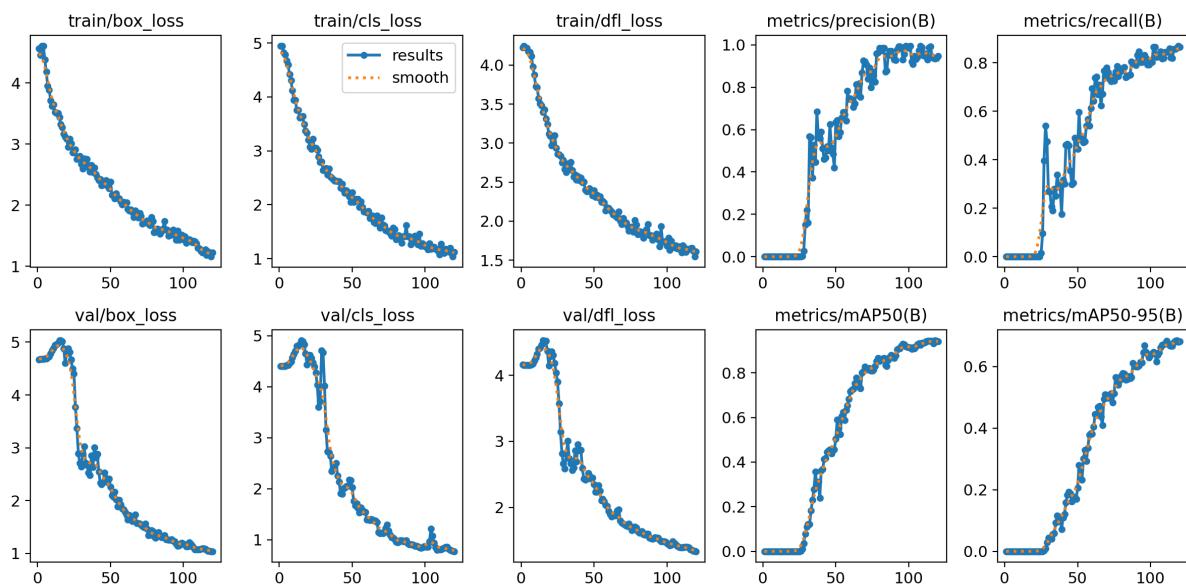
Wyniki zaprezentowane są dwuetapowo. Etap detekcji przez sieć YOLOv8 oraz etap optycznego rozpoznawania znaków poprzez Tesseract v5. W sekcji YOLOv8 omówione zostają zarówno bezpośrednie wyniki działania wytrenowanej sieci, ale także postęp jej uczenia. Zostaje przeprowadzona analiza na zbiorze treningowym, walidacyjnym, oraz sprawdzone zostaje zdolność systemu do generalizacji poprzez zbiór testowy (rzeczywisty).

Etap Tesseract v5 działa na podstawie wyników podanych przez sieć YOLOv8. Wyniki tego etapu są więc też i równocześnie wynikami działania całego zaproponowanego systemu. Rozpatrzone zostają wartości WER, MER oraz CER dla jakości odczytu tekstu. Analizie poddane zostają rozdzielenie na wartości dla poszczególnych kategorii informacji na etykiecie (nadawca, odbiorca, uwagi) oraz na typ etykiety (Inpost, Pocztex, DHL, DPD). Finalnie przeanalizowane są przypadki dobrego oraz słabego działania preprocessingu do silnika Tesseract, aby pokazać możliwe pole do poprawy systemu w przyszłości.

### **4.1. Klasteryzacja tekstu**

Trening sieci przebiegał przez 120 epok uczenia na zbiorze treningowym oraz walidacyjnym. Oba zbiory pochodzą z wygenerowanego przeze mnie zbioru danych. W zbiorze treningowym było 70 etykiet, na każdej widniejącej pozycja uwag, odbiorcy oraz nadawcy paczki. Zbiór walidacyjny składał się z 30 etykiet, również na każdej zawierającej uwagi, odbiorcę oraz nadawcę paczki. W celach analizy przebiegu uczenia generowane są wykresy zawierające metryki na zbiorze treningowym oraz walidacyjnym pokazujące postęp procesu w zależności od epoki. Wykresy automatycznie generują się po skończonym treningu jako bezpośrednią funkcjonalność biblioteki Ultralytics.

Na powyższym wykresach widać przede wszystkim zmniejszającą się poprawę w skuteczności działania modelu z każdą kolejną epoką. Wybrana liczba 120 epok była kompromisem pomiędzy dobrą skutecznością a krótkim czasem treningu. Zarówno straty treningowe jak i

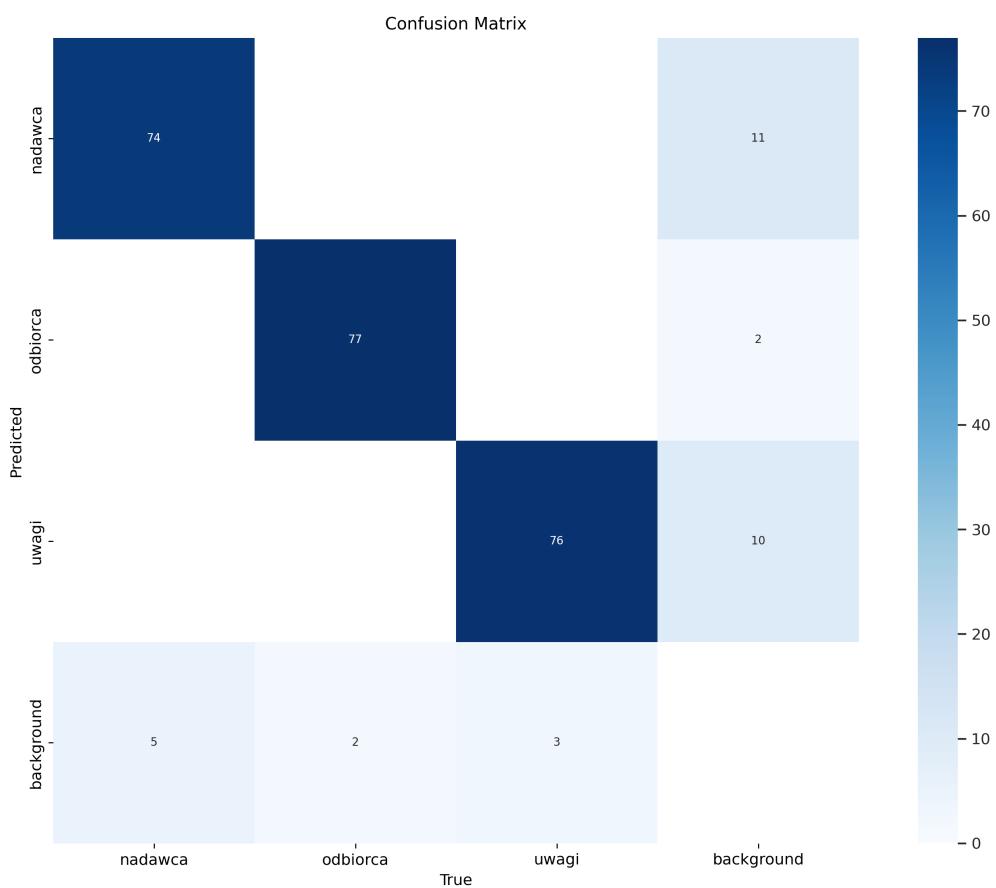


**Rys. 4.1.** Wykresy postępu uczenia sieci YOLOv8. Stworzone za pomocą narzędzi Ultralytics. W pierwszych 3 kolumnach widnieją wykresy strat na zbiorze treningowym (górny wiersz) oraz zbiorze walidacyjnym (dolny wiersz). Kolejne dwie kolumny to obliczone metryki detekcji na zbiorze walidacyjnym (oba wiersze).

walidacyjne zmniejszają się - można tutaj mieć pewność, że sieć faktycznie się uczy. Równocześnie, metryki takie jak precyzaja czy też recall, ulegają definitywnej poprawie w trakcie uczenia. Model ewidentnie działa coraz lepiej w z każdą kolejną epoką.

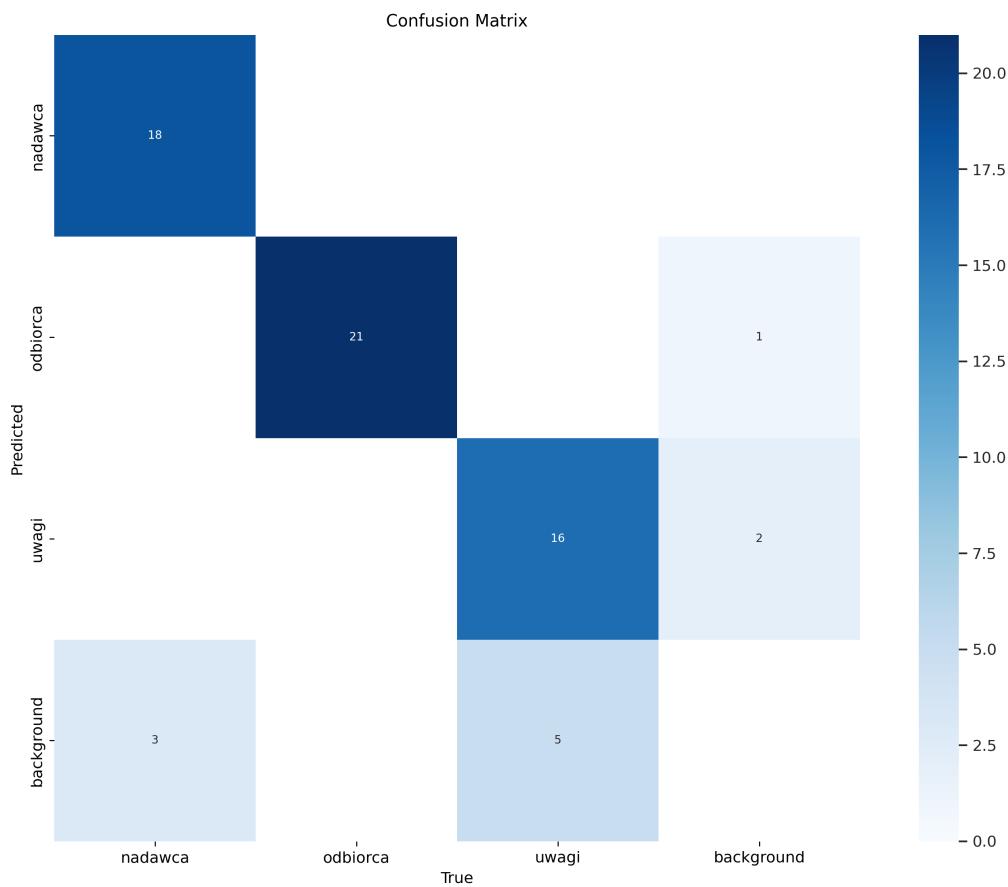
Co jest ważne zaznaczenia to różnica pomiędzy metrykami mAP50 oraz mAP50-95. Metryka gdzie zakładany jest stopień IoU 0.5 jest bliska wartości 0.9, podczas gdy mAP50-95 już tylko 0.7. Świadczy to bezpośrednio o problemach modelu dla dokładnego wykrycia ramek. W idealnym przypadku chciano by, żeby te wartości były do siebie jak najbardziej zbliżone (oraz jak najwyższe). Natomiast tutaj, jeżeli chce się zapewnić większą jakość pojedynczych detekcji (tj. wykryta rama będąca jak najbliżej faktycznej ramki) poświęcone zostanie wtedy dużo strat na precyzyji działania modelu.

Poza modelami postępu uczenia można też uzyskać wykres tablicy kontyngencji. Rzeczywisty wykres dla wytrenowanego modelu przeanalizowany został dla trzech dostępnych zbiorów: treningowego, walidacyjnego, oraz testowego. Dla przypomnienia: zbiór testowy był etykietami rzeczywistymi, pozostałe dwa były to etykiety wygenerowane.



Rys. 4.2. Tabela kontyngencji dla zbioru treningowego.

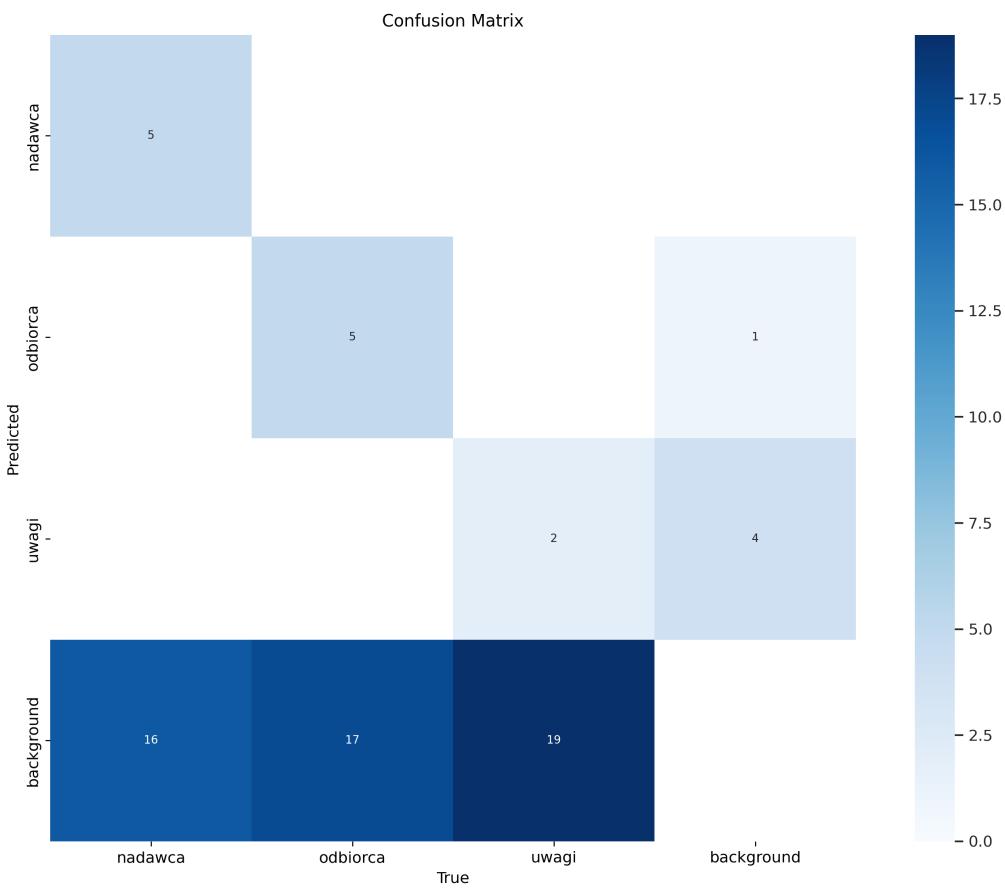
Na zbiorze treningowym model ewidentnie radzi sobie dobrze. Natrafiają się problemy zarówno z nadawcą jak i z uwagami, gdzie model może potraktować tło jako część szukanej klasy. Może to być spowodowane chociażby tym, że bardzo często uwagi mogą być pozostawione puste. Dodatkowo, nadawca na etykiecie czasami zapisany jest znacznie mniejszym drukiem niż odbiorca - takim samym drukiem jak chociażby adres siedziby przewoźnika, który dostarczył paczkę. Mimo niewielkiego rozmiaru zbioru treningowego, uzyskane wyniki detekcji są bardzo satysfakcjonujące. Osiągnięta dokładność świadczy o dobrze przeprowadzonej optymalizacji modelu oraz skuteczności algorytmu detekcji, co sugeruje, że nawet przy ograniczonej liczbie próbek system jest w stanie generalizować wzorce i rozpoznawać obiekty z wysoką precyzją.



Rys. 4.3. Tabela kontyngencji dla zbioru walidacyjnego.

Na zbiorze walidacyjnym sytuacja prezentuje się bardzo podobnie, a nawet korzystniej niż na zbiorze treningowym. Wyniki wskazują na dalszą poprawę skuteczności detekcji modelu, co może świadczyć o jego dobrej zdolności do generalizacji na nowych danych. Model znacznie lepiej poradził sobie tutaj z rozpoznaniem adresu nadawcy, bez utraty skuteczności detekcji odbiorcy. Gorzej z detekcją uwag. Bardzo możliwe, że powodem utraty skuteczności były te same wzgłydy, co wymienione dla tablicy kontyngencji zbioru treningowego. Mimo że kategoria "Uwagi" ma niższy priorytet w procesie logistycznym, kluczowe informacje dla przewoźnika dotyczą przede wszystkim danych o nadawcy i odbiorcy. Z tego powodu można uznać, że model spełnia swoje założenia, ponieważ jego efektywność w detekcji i klasyfikacji tych kluczowych kategorii jest wyższa. Skupienie się na precyzyjnym rozpoznawaniu danych nadawcy i odbiorcy odpowiada na rzeczywiste potrzeby operacyjne przewoźników, gdzie prawidłowa identyfikacja tych informacji jest niezbędna dla prawidłowej realizacji procesu dostawy.

Sytuacja znacząco się pogarsza kiedy spojrzy się na wyniki zbioru testowego. Model po-mimo działania na zbiorach treningowym i walidacyjnym, na zbiorze testowym wykazuje

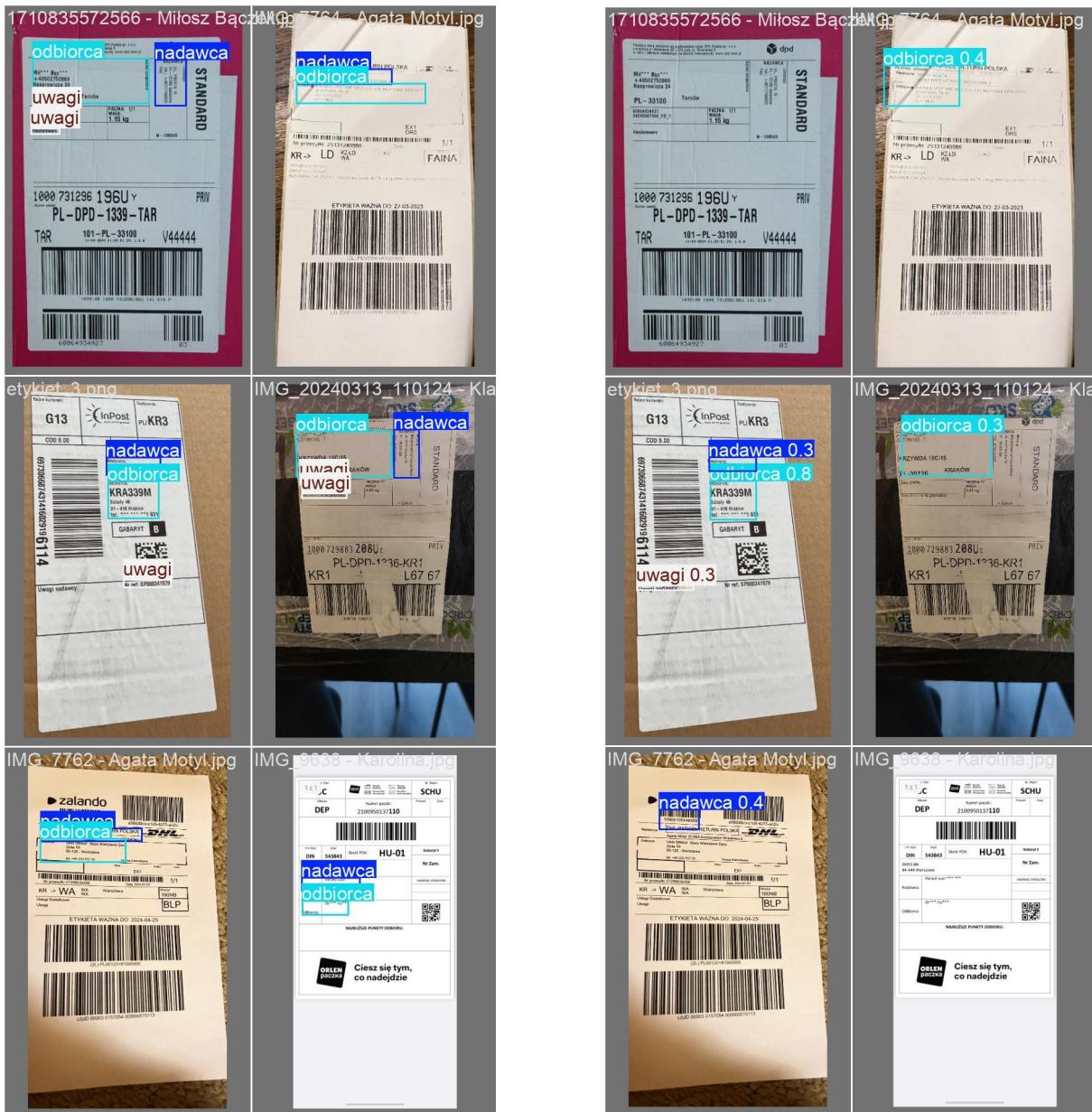


Rys. 4.4. Tabela kontyngencji dla zbioru testowego.

znaczne pogorszenie swojej skuteczności. Model ewidentnie wyuczył się jedynie wzorców dostępnych w zbiorach użytych do treningu, przez co nie generalizuje wystarczająco dobrze. Pokazuje to bezpośrednio, że jako że model działa dla zbiorów treningowego i walidacyjnego, definitywnie możliwa jest detekcja informacji na etykietach (zgodnie z założeniami [5]) w języku polskim. Jednocześnie, aby to osiągnąć potrzebujemy znacznie bardziej zróżnicowanego zbioru treningowego oraz walidacyjnego. Osiągnięcie tego celu na chwilę obecną wykracza poza aktualne możliwości związane z projektem. W niektórych przypadkach sieć YOLO nie potrzebuje dużych zbiorów treningowych, lecz dla tego zastosowania ewidentnie takowy jest potrzebny. Rozwiążaniem problemów sieci wytrenowanej dla rozwijanego systemu jest oczywiście większy zbiór danych.

Jednym z celów mojej pracy było sprawdzenie, czy opracowany model sieci neuronowej będzie w stanie skutecznie rozpoznać i odczytać informacje z etykiet nieznanych mu wcześniej przewoźników, na przykład "Orlen Paczka", mając dostęp jedynie do danych treningowych zebranych dla etykiet innych firm, takich jak Inpost, DHL, DPD czy Pocztex. Tego rodzaju test

ma na celu wykazanie przewagi podejścia wykorzystującego głębokie sieci neuronowe nad tradycyjnymi metodami, gdzie każda specyfikacja etykiety musiałaby być ręcznie wpisana do systemu. Oznaczałoby to konieczność wprowadzenia reguł, które określałyby, gdzie na konkretnej etykiecie znajduje się informacja o nadawcy, odbiorcy i innych istotnych danych. Udane rozpoznanie przez model etykiet spoza zbioru treningowego sugerowałoby, że sieć neuronowa jest w stanie uogólniać i rozpoznawać wzorce, niezależnie od typu etykiety, co znacząco zwiększyłoby elastyczność i skalowalność rozwiązania.



Rys. 4.5. Przykład predykcji systemu na zbiorze testowym. Po lewej stronie obrazy z naniesionymi ramkami, po stronie prawej predykcja systemu.

Rysunek 4.5 prezentuje przykład 6 etykiet ze zbioru testowego. Po prawej stronie widnieją etykiety z naniesionymi detekcjami systemu, po stronie lewej etykiety z ręcznie naniesionymi ramkami do testów. Przede wszystkim należy zwrócić uwagę na etykietę Orlen-paczka (druga kolumna, ostatni wiersz) oraz DPD (pierwsza kolumna, pierwszy wiersz - nazwijmy ją DPD1). Zaczynając od etykiety Orlen-paczka, była to etykieta której system nigdy wcześniej nie widział. Przy małym zbiorze uczącym (jak w rozważanym przypadku) i z wiedzą o działaniu modelu wyciągniętą z poprzednich wykresów, nie jest zaskoczeniem że nowa etykieta nie zostaje rozpoznana. Ciekawszym jest wspomniana wcześniej etykieta DPD1. Pomimo dobrej jakości zdjęcia i podobnego układu danych na etykiecie w stosunku do etykiet używanych w zbiorze uczącym, nie dokonano żadnej detekcji. Etykieta Inpost oraz DHL ewidentnie działa w tym przykładzie najlepiej, co pokrywa się z pozostałymi testami. Z zebranych w trakcie rozwijania projektu etykietach do zbioru testowego etykieta DPD definitelywnie była najtrudniejsza w detekcji, co również widać na obrazie 4.5. Co bezpośrednio z tego wynika, to że model ma znaczący potencjał do poprawnego działania przy zwiększonych zbiorach treningowym i walidacyjnym. YOLOv8 jest tutaj ewidentnie dobrym wyborem.

## 4.2. Optyczne rozpoznawanie znaków

Po uzyskaniu wyników dotyczących detekcji informacji na obrazie etykiety z sieci YOLOv8 uzyskane ramki zostają przekazane dalej. System wycina element obrazu wskazany przez ramkę, oraz zapamiętuje jakiego rodzaju dane mają się w niej znajdować. Przed dokonaniem rozpoznawania znaków systemem Tesseract potrzebuje on odpowiednio przygotowanych danych. W tym celu potrzeba najpierw przeprowadzić preprocessing wyciętych obrazów. Obraz analizowany przez silnik potrzebuje być zbinaryzowany. Binaryzacja zachodzi poprzez użycie biblioteki opencv poprzez dynamiczne progowanie za pomocą tzw. progowania Otsu [46]. Następnie przeprowadzana jest operacja morfologiczna zamknięcia, w celu pozbycia się zakłóceń związanych ze słabą jakością druku. Tak przygotowany wycinek poddawany jest działaniu silnika, którego wynik porównywany jest z faktyczną informacją zawartą na etykiecie. Porównanie dokonane jest poprzez miary WER, MER oraz CER.

**Tabela 4.1.** Wartości WER z podziałem na typ informacji dla zbioru treningowego, zaokrąglone do 3 liczb znaczących. Wartości WER w tabeli zostały wyliczone poprzez zastosowanie wzoru 3.2 na każdej detekcji, uśrednieniu wyników (kolumna 'WER mean') oraz policzeniu odchylenia standardowego (kolumna 'WER std') dla kategorii. Wartości maksymalne WER dla danej kategorii zostały wyliczone zgodnie ze wzorem 3.4. Zostały one następnie uśrednione (kolumna 'Max WER mean') oraz zostało wyliczone ich odchylenie standarowe (kolumna 'Max WER std') dla kategorii.

Kategoria	WER mean	WER std	Max WER mean	Max WER std
Uwagi	0.876	0.342	1.63	0.283
Odbiorca	0.888	0.251	1.14	0.229
Nadawca	0.870	0.134	1.06	0.106

Jak widać z tabeli 4.1 oraz 4.2 zarówno pole "nadawca" jak i "uwagi" mają podobne wartości dla wyliczonych metryk niezależnie od badanego zbioru. Dla zbioru treningowego wyliczone średnie wartości WER są bardzo zbliżone do siebie, jednakowoż wartości ich odchylenia standardowego się od siebie znaczco różnią. Widać, że najmniejszą wariancją w skuteczności detekcji jest tutaj kategoria "nadawcy". Podobna sytuacja jest dla zbioru walidacyjnego - najczęściej pewnych informacji otrzymamy z pola "nadawcy". Dla zbioru walidacyjnego można również zaobserwować znaczące pogorszenie się średniej wartości WER dla "odbiorcy". Wartość WER dla "odbiorcy" w przypadku zbioru walidacyjnego jest tak bliska 1, że równie dobrze magazynier mógłby w tym przypadku wpisać wszystkie dane ręcznie zamiast poprawiać detekcje systemu. W ekstremalnym przypadku, gdy wartość WER wynosi 1 lub więcej, detekcja jest

**Tabela 4.2.** Wartości WER z podziałem na typ informacji dla zbioru walidacyjnego, zaokrąglone do 3 liczb znaczących. Wartości WER w tabeli zostały wyliczone poprzez zastosowanie wzoru 3.2 na każdej detekcji, uśrednieniu wyników (kolumna 'WER mean') oraz policzeniu odchylenia standaryzowanego (kolumna 'WER std') dla kategorii. Wartości maksymalne WER dla danej kategorii zostały wyliczone zgodnie ze wzorem 3.4. Zostały one następnie uśrednione (kolumna 'Max WER mean') oraz zostało wyliczone ich odchylenie standarowe (kolumna 'Max WER std') dla kategorii.

Kategoria	WER mean	WER std	Max WER mean	Max WER std
Uwagi	0.884	0.269	1.61	0.258
Odbiorca	0.983	0.445	1.29	0.464
Nadawca	0.857	0.214	1.09	0.167

na tyle nieprecyzyjna, że liczba operacji wymaganych do osiągnięcia poprawnego wyniku jest równa lub większa niż wpisanie całego tekstu od zera.

Dodatkowo, porównując maksymalne możliwe wartości WER (Word Error Rate) oraz ich odchylenia standaryzowane, można zauważyc, że kategoria "nadawcy" charakteryzuje się najmniejszymi błędami w detekcji. Wartości te są najbardziej stabilne i przewidywalne w porównaniu do innych kategorii, zarówno w zbiorze treningowym, jak i walidacyjnym. To sugeruje, że model dobrze radzi sobie z rozpoznawaniem danych nadawcy, nawet w przypadku większej liczby zmiennych. Dla porównania, inne kategorie, takie jak "odbiorca" czy "uwagi", mają wyższe odchylenia i wymagają więcej edycji w procesie poprawiania detekcji, co może wskazywać na większą złożoność związana z tymi danymi. Można również zaobserwować znaczące pogorszenie się maksymalnych wartości WER dla zbioru walidacyjnego dla kategorii "odbiorcy" podobnie jak przy średniej wartości WER. System wykazuje znaczną zmienność w skuteczności rozpoznawania tej kategorii, co sugeruje konieczność przeprowadzenia dodatkowej analizy problemu w celu zidentyfikowania potencjalnych przyczyn i obszarów wymagających optymalizacji.

**Tabela 4.3.** Wartości CER z podziałem na typ informacji dla zbioru treningowego, zaokrąglone do 3 liczb znaczących. Wartości CER w tabeli zostały wyliczone poprzez zastosowanie wzoru 3.6 na każdej detekcji, uśrednieniu wyników (kolumna 'CER mean') oraz policzeniu odchylenia standardowego (kolumna 'CER std') dla kategorii. Wartości maksymalne CER dla danej kategorii zostały wyliczone zgodnie ze wzorem 3.7. Zostały one następnie uśrednione (kolumna 'Max CER mean') oraz zostało wyliczone ich odchylenie standarowe (kolumna 'Max CER std') dla kategorii.

Kategoria	CER mean	CER std	Max CER mean	Max CER std
Uwagi	0.876	0.342	1.40	0.251
Odbiorca	0.888	0.251	1.05	0.178
Nadawca	0.870	0.134	1.03	0.051

Jak widać z tabel 4.3 oraz 4.4 wyliczone wartości CER są bardzo zbliżone do wartości WER - wręcz identyczne dla wyznaczonych średnich i odchyleń standardowych. Detekcje różnią się natomiast wartościami maksymalnymi wyliczonymi dla badanych przykładów. Wartości CER mają mniejsze wartości maksymalne porównawczo do wartości WER. Ze względu na ich potencjalnie mniejsze wartości maksymalne, miary CER mogą potencjalnie przekazywać więcej informacji w porównywaniu iteracyjnym systemu ze względu na mniejszy zakres możliwych wartości miary większych od 1. Dla pojedynczej wersji systemu porównanie CER oraz WER nie przekazuje przez to wiele więcej informacji.

**Tabela 4.4.** Wartości CER z podziałem na typ informacji dla zbiorów walidacyjnego, zaokrąglone do 3 liczb znaczących. Wartości CER w tabeli zostały wyliczone poprzez zastosowanie wzoru 3.6 na każdej detekcji, uśrednieniu wyników (kolumna 'CER mean') oraz policzeniu odchylenia standardowego (kolumna 'CER std') dla kategorii. Wartości maksymalne CER dla danej kategorii zostały wyliczone zgodnie ze wzorem 3.7. Zostały one następnie uśrednione (kolumna 'Max CER mean') oraz zostało wyliczone ich odchylenie standarowe (kolumna 'Max CER std') dla kategorii.

Kategoria	CER mean	CER std	Max CER mean	Max CER std
Uwagi	0.884	0.269	1.43	0.269
Odbiorca	0.983	0.445	1.18	0.382
Nadawca	0.857	0.214	1.04	0.0733

**Tabela 4.5.** Wartości MER z podziałem na typ informacji dla zbioru treningowego, zaokrąglone do 3 liczb znaczących. Wartości MER w tabeli zostały wyliczone poprzez zastosowanie wzoru 3.5 na każdej detekcji, uśrednieniu wyników (kolumna 'MER mean') oraz policzeniu odchylenia standardowego (kolumna 'MER std') dla kategorii.

Kategoria	MER mean	MER std
Uwagi	0.686	0.185
Odbiorca	0.817	0.140
Nadawca	0.847	0.132

Porównując wyniki WER do wyników MER (tj. tabeli 4.5 dla danych treningowych oraz 4.6 dla danych walidacyjnych) można zauważyc różnicę w otrzymanej informacji. Wartości MER oraz WER często są ze sobą tożsame, lecz w tym przypadku tak nie jest - szczególnie przy detekcji kategorii "uwagi". Dzięki MER można zaobserwować, że zarówno dla zbioru treningowego jak i walidacyjnego wykryte słowo dla kategorii uwag ma znacznie mniejszą szansę bycia wykryte błędnie od pozostałych kategorii. Na podstawie analizy WER (Word Error Rate) można wyciągnąć wnioski, że w przypadku tej kategorii system rzadziej identyfikuje słowa w porównaniu z innymi kategoriami. Jednakże, kiedy już wykryje słowo, istnieje większe prawdopodobieństwo, że detekcja jest poprawna. Sugeruje to, że problemem może być trudność w wykrywaniu samej obecności słów, a niekoniecznie ich błędna klasyfikacja, co wskazuje na potencjalne wyzwania związane z jakością obrazu, zakłóceniami wizualnymi lub specyfiką danych wejściowych dla tej konkretnej kategorii.

**Tabela 4.6.** Wartości MER z podziałem na typ informacji dla zbioru walidacyjnego, zaokrąglone do 3 liczb znaczących. Wartości MER w tabeli zostały wyliczone poprzez zastosowanie wzoru 3.5 na każdej detekcji, uśrednieniu wyników (kolumna 'MER mean') oraz policzeniu odchylenia standardowego (kolumna 'MER std') dla kategorii.

Kategoria	MER mean	MER std
Uwagi	0.737	0.185
Odbiorca	0.794	0.136
Nadawca	0.808	0.189

Jako że miara MER daje nam informacje o utracie informacji lepszą dla analizy przez system komputerowy niż chociażby WER (najlepsze dla systemu gdzie na końcu ktoś może poprawić ręcznie detekcje) czy też CER, może to prowadzić do wniosków że ostatni krok całości systemu - rozpoznanie paczki na podstawie informacji odczytanych z tekstu na etykiecie - powinno również być zautomatyzowane. Dopiero na samym końcu magazynier mógłby np. potwierdzić poprawne rozpoznanie przesyłki. Dodatkowo bliskie podobieństwo pomiędzy wartościami CER oraz WER sugeruje, że na jedno niepoprawne słowo przypadał około jedna złe rozpoznany znak. Przez to również wydaje się poprawne zastosowanie algorytmu jako ostatniego kroku systemu, zamiast elementu ludzkiego. Literówki mogą zostać łatwo rozwiązane przez system informatyczny, np. poprzez walidację adresu z chociażby aplikacją zawierającą w sobie aktualne polskie adresy administracyjne.

**Tabela 4.7.** Wyliczone wartości WER dla zbioru treningowego z podziałem na typ etykiety, zaokrąglone do 3 liczb znaczących. Tabela wyliczona tym samym sposobem co tabela 4.1, zamiast podziału na kategorie zastosowano podział na przewoźnika.

Przewoźnik	WER mean	WER std	Max WER mean	Max WER std
InPost	0.878	0.254	1.32	0.374
Pocztex	0.873	0.243	1.28	0.313
DPD	0.866	0.476	1.29	0.337
DHL	0.889	0.155	1.20	0.209

Wiele też będzie zależeć od typu etykiety, która jest analizowana. W tabelach 4.7 oraz 4.8 można zobaczyć wyliczone wyniki wartości WER w zależności od typu etykiety jaką analizował system. Z ciekawszych obserwacji można wskazać coś czego można się było spodziewać

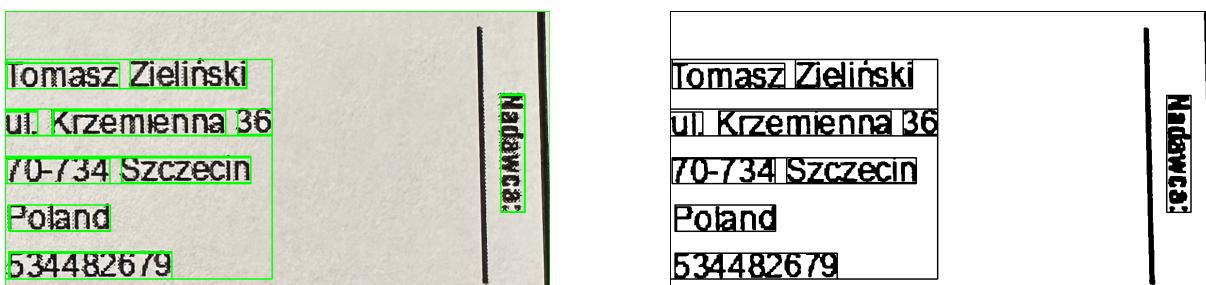
**Tabela 4.8.** Wyliczone wartości WER dla zbioru walidacyjnego, z podziałem na typ etykiety, zaokrąglone do 3 liczb znaczących. Tabela wyliczona tym samym sposobem co tabela 4.2, zamiast podziału na kategorie zastosowano podział na przewoźnika.

Przewoźnik	WER mean	WER std	Max WER mean	Max WER std
InPost	0.899	0.241	1.38	0.366
Pocztex	0.875	0.147	1.20	0.205
DPD	1.28	0.642	1.74	0.562
DHL	0.729	0.186	1.13	0.165

już po wynikach cząstkowych z sieci YOLOv8 - najgorzej wypada tutaj etykieta DPD. Wartości średniej WER są do siebie zbliżone dla zbioru treningowego, to fakt. Znacząca różnica jest natomiast widoczna dla zbioru walidacyjnego. Czy to zła jakość druku czy to nieregularne ułożenie tekstu na etykiecie (szczególnie dla "odbiorcy") wartość WER dla zbioru walidacyjnego dla etykiety DPD jest znacznie większa od 1. Oznacza to, że bezpośrednio szybciej będzie magazynierowi ręcznie wpisać tekst, niż poprawiać detekcję systemu. Pozostałe etykiety w zbiorze walidacyjnym wykazują bardzo podobną zgodność co do jakości detekcji - około 0.85 WER - oraz podobne odchylenie standardowe co w zbiorze treningowym.

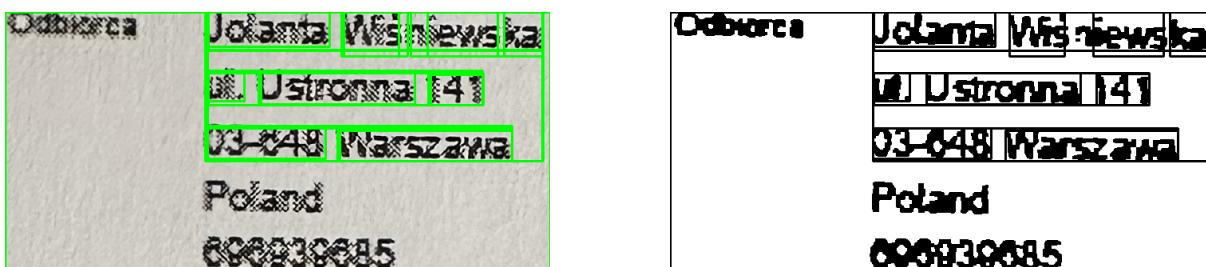
Żeby zrozumieć gdzie dokładnie tkwi problem z działaniem systemu (poza oczywistą poprawą działania sieci poprzez większy zbiór uczący) należy bezpośrednio spojrzeć na wycinki które otrzymuje Tesseract przed i po preprocessingu. Na rysunku 4.6 można zobaczyć przykład kiedy to tekst na wycinku jest dobrej jakości. Druk nie uległ uszkodzeniu, czcionka jest dużego rozmiaru, brak zakłóceń na obrazie. Można zauważyć, że każde słowo zostało wychwycone bez problemu przez system. Kolejnym krokiem jest zobaczenie czy dla tego konkretnego przykładu wychwycone poprawnie słowo, jest również poprawne. Dla tego przykładu błędy były na poziomie pojedynczych liter - "Zelinski" zamiast "Zieliński" oraz "Polana" zamiast "Poland".

Bardziej zaawansowanym przypadkiem jest rozpoznanie tekstu kiedy jakość druku ulega pogorszeniu. Na rysunku 4.7 już na obrazie przed obróbką (po stronie lewej) ciężko jest odczytać tekst. Nie jest to jednak niemożliwe dla "ludzkiego eksperta". Skoro dla ludzkiego eksperta zadanie już jest utrudnione, można przewidywać, że dla systemu komputerowego zadanie również będzie mieć wyższy stopień trudności. Tak również dzieje się tutaj. Po zastosowaniu tego samego procesu przetwarzania danych przed analizą silnikiem Tesseract widać, że jakość tekstu dodatkowo się pogarsza. Kiedy ludzki ekspert popatrzy na obraz po obróbce (po prawej) odczyt tekstu jest znacznie bardziej skomplikowany. Litery zlewają się w całość, cyfry tracą kształt.



Rys. 4.6. Przykład działania Tesseract na czytelnym, dobrej jakości tekście.

Po stronie lewej wycinek obrazu na podstawie wyników z sieci YOLOv8. Po stronie lewej ten sam wycinek po operacjach preprocessingu. Na obydwu obrazach naniesione zostały ramki wykrytego tekstu przez Tesseract.



Rys. 4.7. Przykład działania Tesseract na nieczytelnym, słabej jakości tekście.

Po stronie lewej wycinek obrazu na podstawie wyników z sieci YOLOv8. Po stronie lewej ten sam wycinek po operacjach preprocessingu. Na obydwu obrazach naniesione zostały ramki wykrytego tekstu przez Tesseract.

W przypadkach gdy jakość druku jest gorsza, a czcionka użyta na etykiecie jest za mała, wtedy cały system napotyka na największe przeszkody. Co więcej, spadek na jakości jest o tyle duży, że wydaje się, że samą pracą nad obróbką zdjęć etykiet ze słabą jakością druku działanie systemu zyskałoby znacznie bardziej na jakości, niż nawet wyuczenie lepszej sieci do detekcji informacji na zdjęciach rzeczowych etykiet. Wyzwanie słabej jakości druku będzie o tyle bardziej powszechnne, o ile rosnące nadal będą na popularności serwisy oferujące automatyczną generację etykiet kurierskich. Chociażby popularny serwis Vinted generuje i wysyła etykietę osobie sprzedającej (która to też później wysyła paczkę - jest jej nadawcą). Nadawca następnie samodzielnie drukuje wygenerowaną etykietę. Większość klientów zrobi to na własnej drukarce. Domowa drukarka będzie gorszej jakości, niż przystosowana do tego drukarka etykiet kurierskich. Problem będzie narastać z biegiem czasu, jako że tego typu rozwiązania dla klientów są coraz bardziej popularne i powszechnne.

Dzięki przeprowadzonemu procesowi tworzenia projektu można było bezpośrednio zidentyfikować wąskie gardło całego rozwiązania - jakość druku. Rozwiązanie tego zagadnienia pozwoli na znaczącą poprawę działania całego systemu w przyszłości.

Rozpoznanie tego wąskiego gardła jest ważnym krokiem w rozwoju systemu, ponieważ pozwala na lepsze zrozumienie, w których obszarach model wymaga dalszego dopracowania. Pozwoli to także na bardziej precyzyjne ukierunkowanie przyszłych działań, takich jak np. dostosowanie procesu trenowania sieci, wybór odpowiednich zbiorów danych, czy optymalizacja preprocesingu tekstu. Wnioski z takich analiz mogą stać się podstawą do dalszych badań oraz prowadzić do bardziej zaawansowanych metod, które będą w stanie poradzić sobie z trudniejszymi przypadkami, jak np. rozpoznawanie danych odbiorcy czy uwag, które są bardziej zróżnicowane i wymagają większej elastyczności modelu.



## **5. Podsumowanie oraz wnioski**

Wyniki jednoznacznie wskazują na wysoki potencjał zastosowanego systemu. Dzięki przeprowadzonym eksperymentom wskazane zostały także dwa potrzebne rozwiązania do badanego problemu w celu zmaksymalizowania wydajności proponowanego projektu. Jeżeli zaproponowany system zostanie rozłożony na jego dwie części składowe - klasteryzację oraz odczyt tekstu - można bezpośrednio nazwać rzeczone dwa rozwiązania. Dla klasteryzacji tekstu należy uzykać większy i bardziej zróżnicowany zbiór danych. Dla detekcji tekstu należy opracować rozwiązanie zajmujące się wyłącznie etykietami z niską jakością DPI oraz małą czcionką. Dzięki przekrojowemu podejściu do zagadnienia które chciano zrealizować, można bezpośrednio i z pewnością powiedzieć jakie rozwiązania są potrzebne do pełnego działania systemu.

Jeżeli skupiono by się w trakcie tworzenia projektu systemu na pojedynczej jego części, uważam, że wymienione wyżej wyzwania zostałyby w pełni zrealizowane. Zaczynając od problemów z jakością druku, czyli problemami w rozwiązaniach OCR. Przy końcu realizacji projektowej części pracy magisterskiej pozostawiono czas na próby poprawy jakości detekcji poprzez eksperymenty z pre-processingiem obrazów. Dla silnika Tesseract zalecana jest dynamiczna binaryzacja obrazu, oraz operacje morfologiczne w zależności od jakości druku. Problem tutaj polegał na tym, że inne operacje morfologiczne działały lepiej na małej czcionce, a inne na dużej. Rozwiązaniem tutaj mogłoby być utworzenie klasyfikatora, który aproksymowałby wielkość czcionki i na tej podstawie wybierał operacje. Przy założeniu że etykieta zawsze wypełnia 80-90% zdjęcia, które podlega analizie, nawet rozwiązanie statyczne jest wysoce prawdopodobne zadziałać.

Zagadnienie jest znacznie bardziej złożone w przypadku poprawy działania klasteryzacji tekstu - czyli sieci YOLOv8. Nie tylko zbiór danych był ewidentnie za mały, ale też za mało zróżnicowany. Bez większego i bardziej zróżnicowanego zbioru sieć będzie się przeuczać, przez co nie będzie generalizować. W niektórych przypadkach sieć YOLO nie potrzebuje dużych zbiorów treningowych, lecz dla tego zastosowania ewidentnie takowy jest potrzebny.

Zbiór ten można pozyskać na dwa sposoby - drogą ankietową, lub też poprzez generację. Jak to można spostrzec ankietowani nie są chętni do dobrowolnego udostępniania swoich danych

osobowych dostępnych na etykietach paczek. Pozostaje tylko generacja. Jeżeli potrzeba by podejść do zagadnienia powtórnie, skupiono by się tylko i wyłącznie na zagadnieniu klasteryzacji. Pozwoliłoby to utworzyć generator danych, który tworzyłby nie tylko etykiety, ale i jednocześnie tworzył metodami generatywnymi "zdjęcia" tych etykiet, razem z gotowymi ramkami. Takie narzędzie pozwoliłoby skalować zbiór danych wedle potrzeb, bez wymagania drukowania oraz fizycznego robienia zdjęć. Dla skali 100 etykiet którą wybrano dla projektu zrobienie zdjęć ręcznie nie było aż takim wyzwaniem. Niemniej jednak dla skali np. 10 tys zdjęć podejście manualne w żaden sposób nie wydaje się być możliwe do zrealizowania.

Po za przygotowaniem proponowanych rozwiązań całość systemu mogłaby zostać opakowana w przyjazne GUI w celach testowych dla kurierów, czy też może nawet dla regularnych klientów usług kurierskich. Osobiście często zdarza mi się coś odsyłać do nadawcy. Chociażby możliwość bezpośredniego odczytania danych poprzedniego nadawcy paczki przyspieszyłaby cały proces. Jeżeli nawet nie w celach testowych dla użytkowników końcowych, GUI definitelywnie poprawiłoby jakość oraz tempo kolejnych eksperymentów związanych z działaniem systemu. Proces od generacji zdjęcia etykiety aż do finalnych wyników powinien być uregulowany i możliwie poddawany edycji z poziomu GUI.

Proponowany system rzeczywiście posiada ogromny potencjał do dalszego rozwoju. Zagadnienia, które zostały poruszone w tym projekcie, oferują liczne możliwości rozszerzenia zarówno w kierunku horyzontalnym, jak i wertykalnym. Rozwój horyzontalny pozwoli na wprowadzenie nowych typów danych do systemu, na przykład poprzez dodanie różnych typów etykiet, czy też uwzględnienie rozmaitych języków i formatów, co może znacząco zwiększyć zakres i użyteczność systemu. Rozwój wertykalny może obejmować bardziej zaawansowane algorytmy przetwarzania danych, poprawę precyzji w rozpoznawaniu tekstu czy automatyzację procesu detekcji w bardziej skomplikowanych przypadkach.

Kolejne kroki w rozwoju projektu mogą obejmować między innymi integrację z innymi systemami, co umożliwi wymianę danych w czasie rzeczywistym, jak również rozwój interfejsów użytkownika, które pozwolą na jeszcze łatwiejsze korzystanie z narzędzi analitycznych. Proponowany system jest tylko początkiem, a przyszłość przynosi kolejne wyzwania, które z pewnością zostaną pokonane dzięki dalszym badaniom i implementacjom.

## **A. Załączniki**

## A.1. Ankieta zbierania danych rzeczywistych

11.08.2024, 20:19

Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"

### Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"

Nazywam się Michał Motyl i obecnie jestem w trakcie studiów magisterskich na Automatyce i Robotyce ze specjalizacją Informatyka w Sterowaniu i Zarządzaniu na Akademii Górnictwa i Hutniczej w Krakowie. Tematyka tworzonej przez mnie pracy magisterskiej to "Opracowanie systemu do analizy informacji na etykietach paczek produktów". W mojej pracy chcę przede wszystkim skupić się na **etykietach paczek kurierskich**. Etykieta kurierska to rodzaj identyfikatora, który jest umieszczany na zewnętrznej powierzchni paczki lub przesyłki. Jest to zazwyczaj naklejka zawierająca różnorodne informacje związane z przesyłką oraz niezbędne dane dla firm kurierskich i operatorów logistycznych.

Głównym utrudnieniem tego procesu, jest brak publicznej dostępności zbiorów danych rzeczywistych etykiet. Z tego względu chciałbym gorąco zachęcić do wypełnienia poniższej ankiety w celu stworzeniu opisanego zbioru danych etykiet paczek kurierskich. **Nadsyłać należy tylko etykiety, które są w pełni opisane w języku polskim.**

Ankieta polega na wrzucenia czytelnego zdjęcia w formacie ".jpg" które przedstawia opisywaną później etykietę paczki kurierskiej. W kolejnych krokach etykietę można odpowiednio opisać wedle wskazanych przeze mnie kroków, lecz nie jest to konieczne jeżeli komuś zależy na czasie. Jednokrotne opisanie etykiety powinno zajść około 10 minut.

Jedno wypełnienie ankiety to wysłanie i ewentualne opisanie jednej etykiety. Zachęcam do wypełniania ankiety wielokrotnie.

Zbiór danych zostanie wykorzystany tylko i wyłącznie do testowania tworzonego przeze mnie rozwiązania. Nie zostanie od nigdzie potem opublikowany, ani nigdzie potem przetwarzany w innych celach. Jeżeli zajdzie potrzeba pokazania fragmentu zbioru w pracy naukowej, w pierwszej kolejności dane zostaną poddane anonimizacji.

*Wypełniając ankietę wypełniająca osoba daje zgodę na wykorzystanie podanych przez nią danych jedynie w celach naukowych ograniczających się do pracy magisterskiej mojego autorstwa.*

Wypełniając ankietę wyrażam zgodę na przetwarzanie moich danych osobowych podanych w

11.08.2024, 20:19

Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"

tej ankiecie w celu opracowania systemu do analizy informacji na etykietach paczek przez Michała Motyla.

Zapoznaj się z klauzulą informacyjną.

\* Indicates required question

### Klauzula informacyjna

Informuję, że:

1. administratorem Pani/Pana danych osobowych jest Michał Motyl, zamieszkały: ul. Widokowa 3, 32-065 Krzeszowice (dalej: Administrator), który przetwarza Pani/Pana dane osobowe;
2. Pani/Pana dane osobowe przetwarzane będą w celu przygotowania danych wejściowych do tworzonej przeze mnie pracy magisterskiej to "Opracowanie systemu do analizy informacji na

etykietach paczek produktów" i nie będą udostępniane innym odbiorcom;

3. podstawą przetwarzania Pani/Pana danych osobowych jest art. 6 ust. 1 lit. a)

Rozporządzenie Parlamentu Europejskiego i Rady Unii Europejskiej z 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/W (ogólne rozporządzenie o ochronie danych);

4. posiada Pani/Pan prawo do:

\* żądania od Administratora dostępu do swoich danych osobowych, ich sprostowania, usunięcia lub ograniczenia przetwarzania danych osobowych,  
\* wniesienia sprzeciwu wobec takiego przetwarzania,  
\* przenoszenia danych,  
\* wniesienia skargi do organu nadzorczego,  
\* cofnięcia zgody na przetwarzanie danych osobowych.

5. Pani/Pana dane osobowe nie podlegają zautomatyzowanemu podejmowaniu decyzji, w tym profilowaniu;

6. Pani/Pana dane osobowe będą przetwarzane z użyciem narzędzi dostarczanych przez Google.

1. Zapoznałem/Zapoznałam się z Klauzulą Informacyjną \*

Mark only one oval.

Tak

Nie

### Informacje ogólne

11.08.2024, 20:19

Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"

2. Proszę wrzucić zdjęcie etykiety paczki kurierskiej, którą będzie się opisywać  
(wymaga bycia zalogowania do konta google) \*

Files submitted:

3. Czy etykieta **napewno** jest w języku polskim? \*

*Mark only one oval.*

- Tak  
 Nie

4. Jakiego przewoźnika jest opisywana przez Ciebie etykieta? \*

*Mark only one oval.*

- Inpost  
 DPD  
 DHL  
 UPS  
 Pocztex  
 Fedex  
 GLS  
 Orlen PACZKA  
 Other: \_\_\_\_\_

#### Wybór opisu

Proszę zdecydować czy chcą Państwo ręcznie opisać etykietę.

11.08.2024, 20:19

Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"

## 5. Wybór: \*

*Mark only one oval.* Chcę opisać etykietę Nie chcę opisywać etykiety - zakończ ankietę *Skip to section 5 (Zakończenie)***Opis etykiety**

Poniżej proszę opisać odpowiednie dane widniejące na wcześniej przesłanym przez Państwa zdjeciu etykiety.

Proszę największą uwagę zwrócić na **wpisywanie danych dokładnie w taki sam sposób jak na etykiecie**. Nawet jeżeli na etykiecie widnieje literówka, proszę wprowadzić dane uwzględniając ją.

Jeżeli na etykiecie widnieją dane po części ukryte, tj. zamiast pełnego imienia i nazwiska bądź też numeru telefonu widnieją tylko odpowiedniki z zakodowanymi literami, proszę wpisać do ankiety dane zakodowany w taki sam sposób. Przykład: na paczce zamiast odbiorcy "Michał Motyl nr tel 123 456 789" widnieje "Mi\*\*\*\* M\*\*\*\* nr tel 1\*\* \*\*\* \*89" w ankiecie proszę wpisać wersję "Mi\*\*\*\* M\*\*\*\* nr tel 1\*\* \*\*\* \*89".

**Jeżeli jakieś dane nie ma na etykiecie lub zwyczajnie nie jest się w stanie jej znaleźć, proszę wpisać znak równości "=" w miejscu brakującej danej.**

11.08.2024, 20:19

Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"

## 6. Numer przewozowy \*

Numer przewozowy (tożsamy z numerem paczki czy też numerem przesyłki) to kilkunastocyfrowy numer wykorzystywany przez przewoźnika, żeby zidentyfikować paczkę. Dla przewoźnika jest to najważniejsza informacja na etykiecie paczki, gdyż pozwala on na bezpośrednie odnalezienie danych o paczce w systemie, bez konieczności odczytu ich bezpośrednio z etykiety. Dostawca jest go w stanie odczytać automatycznie poprzez użycie skanera kodów kreskowych lub też kodów QR (zależnie od paczki). Niemniej jednak numer przewozowy **musi się znaleźć w formie pisanej** na etykiecie. Zazwyczaj możemy go znaleźć w okolicach kodu QR, bądź też kodu kreskowego.

Przykładowe formaty listu przewozowego:

- 11 cyfr: np. 15467388748,
- JJD: np. JJD149020000300000000173,
- 12 cyfr: np. 422891590640,
- 18 cyfr: np. 00340433982034779798,
- 2 litery+9 cyfr+2 litery: np. CY730190157DE.

Jeżeli numer listu przewozowego jest zamazany bądź też któryś jego elementy są nieczytelne, proszę wprowadzić te elementy jako gwiazdki (\*) np. w numerze "CD12345EF" cyfra 4 jest zamazana, a litera F jest pomazana długopisem - wtedy należy wpisać numer jako "CD123\*5E\*"

Jeżeli z jakiegoś powodu numer ten jest zupełnie nieczytelny, bądź też nie jest się go w stanie znaleźć, proszę w odpowiedzi wpisać znak równości (=).

---

---

---

---

11.08.2024, 20:19

Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"

**7. Nadawca \***

Jeżeli dane nadawcy mają więcej niż jedną linijkę na etykiecie, proszę wpisać całość w odpowiedzi a linijki oddzielić podwójnym slashem (//) .

W przypadku braku danych o nadawcy na etykiecie, proszę wpisać znak równości w odpowiedzi (=).

---

---

---

---

---

**8. Odbiorca \***

Jeżeli dane odbiorcy mają więcej niż jedną linijkę na etykiecie, proszę wpisać całość w odpowiedzi a linijki oddzielić podwójnym slashem (//) .

W przypadku braku danych o nadawcy na etykiecie, proszę wpisać znak równości w odpowiedzi (=).

---

---

---

---

---

11.08.2024, 20:19

Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"

**9. Adres doręczenia/kod paczkomatu \***

Czasami niektórzy przewoźnicy dodatkowo podają adres doręczenia, np. jeżeli adres odbiorcy różni się od adresu gdzie paczka została wysłana. Niektórzy przewoźnicy nawet przekazują jedynie adres doręczenia, bez podawania danych odbiorcy na etykiecie. Na etykiecie adres doręczenia powinien być jasno opisany słowami "Adres doręczenia" w przypadku gdzie paczka była doręczona do czystej rąk, lub np. może to być **kod paczkomatu do którego paczka została doręczona**.

Jeżeli adres doręczenia ma więcej niż jedną linijkę, proszę wpisać całość w odpowiedzi a linijki oddzielić podwójnym slashem (//) .

Jeżeli takowego na etykiecie nie ma, proszę wpisać w odpowiedzi znak równości (=).

---

---

---

---

**10. Numer do śledzenia paczki online \***

Niektórzy przewoźnicy poza numerem przewozowym podają na etykiecie również numer do śledzenia paczki w ich serwisach internetowych. Numer ten wcale nie musi być tożsamy z numerem przewozowym (np. GLS podaje ten właśnie numer osobno).

W tym pytaniu interesuje mnie jedynie sytuacja kiedy numer śledzenia jest inny, niż numer przewozowy.

Jeżeli na paczce nie ma jasno nazwanego numeru do śledzenia paczki online, lub zwyczajnie nie jest w stanie się takiego znaleźć na etykiecie paczki, najpewniej oznacza to że numer przewozowy jest tożsamy z numerem do śledzenia paczki online. W takiej sytuacji proszę wpisać w odpowiedzi znak równości (=).

---

---

---

---

11.08.2024, 20:19

Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"

**11. Adres przewoźnika \***

List przewozowy powinien też zawierać nazwę i adres przewoźnika, tak aby klienci wiedzieli, gdzie zgłaszać ewentualne pytania i reklamacje. Powinien on być czytelnie opisany na etykiecie. Dane takie też są wymagane ze względu na przepisy RODO - w końcu są to dane administratora danych osobowych widocznych na etykiecie.

Jeżeli adres przewoźnika ma więcej niż jedną linijkę, proszę wpisać całość w odpowiedzi a linijki oddzielić podwójnym slashem (//) .

Jeżeli takowego na etykiecie nie ma, proszę wpisać w odpowiedzi znak równości (=).

---

---

---

---

**12. Waga \***

Na paczce może znaleźć się waga przesyłki. Podana może być waga zawartości paczki, oraz waga paczki razem z opakowaniem w którym została przesłana - czyli na paczce mogą widnieć dwie wartości wagi.

Jeżeli na paczce widnieje tylko jedna wartość wagi, proszę ją wpisać poniżej.

W przypadku jeżeli na paczce znajdują dwie wartości, to można mieć pewność że wartość mniejsza jest faktyczną wagą zawartości przesyłki. **Tę wagę proszę wpisać poniżej.**

**Wagę proszę wpisywać w formie dokładnie tak jak na etykiecie (np. "1 kg" czyli wartość oraz jednostka).**

Jeżeli takowego na etykiecie nie ma, proszę wpisać w odpowiedzi znak równości (=).

---

---

---

---

11.08.2024, 20:19

Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"

**13. Gabaryt \***

Na niektórych etykietach widnieje również informacja dotycząca gabarytu przesyłki. Może to być informacja słowna bądź też kategoria, którą symbolizuje litera.

Jeżeli gabaryt na etykiecie określony jest jako kategoria symbolicznie, np. "A", proszę poniżej wpisać "A".

Jeżeli gabaryt na etykiecie określony jest słownie, np. "mały gabaryt", proszę poniżej wpisać "mały gabaryt"

Jeżeli na etykiecie nigdzie nie ma jasno opisanego pola z kategorią gabarytu, proszę wpisać w odpowiedzi znak równości (=).

---

---

---

---

**14. Usługi dodatkowe \***

Czasami też na etykiecie można odczytać usługi dodatkowe jakie były wykupione dla danej przesyłki. Takowymi mogą być np. ubezpieczenie paczki do jakiejś wartości, opłata za ostrożne obchodzenie się z paczką czy też przesyłka priorytetowa. Takich usług może być naturalnie więcej niż jedna.

Jeżeli usług dodatkowych jest więcej niż jedna, proszę wpisać je wszystkie, oddzielając każdą podwójnym slashem (//) .

Jeżeli nie ma opisanych usług dodatkowych na etykiecie, proszę wpisać w odpowiedzi znak równości (=).

---

---

---

---

11.08.2024, 20:19

Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"

**15. Uwagi nadawcy \***

W większości etykiet jest miejsce na przekazanie uwag, które nadawca miał możliwość ręcznie wpisać do systemu przed nadaniem paczki. Zazwyczaj są one widoczne w polu opisanyim "Uwagi". Mogą to być np. "wejście od strony ulicy X" lub "prośba o zostawienie paczki na recepcji". Bardzo często jednak jest ono pozostawione puste, mało kto faktycznie używa tej funkcjonalności.

Jeżeli uwagi nadawcy są dłuższe, i zajmują pare linijek, proszę wpisać je oddzielając linijki podwójnym slashem (//).

Jeżeli na etykiecie nie ma uwag nadawcy, proszę wpisać w odpowiedzi znak równości (=).

---

---

---

---

**16. Data nadania (w formacie DD-MM-YYYY) \***

Na etykiecie rzadziej widnieje data nadania danej paczki, lub też data wydrukowania etykiety. Jeżeli takowa informacja znajduje się na paczce powinna być ona jasno opisana.

Jeżeli na etykiecie znajduje się jasno opisana data nadania paczki, proszę wpisać ją poniżej **w formacie DD-MM-YYYY czyli dzień, miesiąc oraz rok oddzielone znakiem "-".**

---

---

---

---

11.08.2024, 20:19

Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"

**17. Opis zawartości \***

Finalnie, często na etykiecie pojawić się może krótki opis tego co paczka zawiera. Np. mogą to być "kosmetyki", "sprzęt agd" czy też chociażby bardziej szczegółowo "przedmioty łatwo tłukące się i szkło".

Jeżeli na etykiecie nie widnieje nigdzie opis zawartości paczki, proszę w poniższym polu wpisać znak równości (=).

---

---

---

---

---

**Zakończenie**

Bardzo dziękuję za wypełnienie ankiety. Proszę wcisnąć submit w celu jej wysłania.

---

This content is neither created nor endorsed by Google.

Google Forms

11.08.2024, 20:19

Zbiór danych: "Opracowanie systemu do analizy informacji na etykietach paczek produktów"



## Bibliografia

- [1] Sungho Suh, Paul Lukowicz i Yong Oh Lee. „Fusion of global-local features for image quality inspection of shipping label”. W: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, s. 2643–2649.
- [2] Coen Jannsen, Daniel Mullen i in. *GS1 Logistic Label Guideline*. 2019.
- [3] Jianhua Zheng i in. „EHFP-GAN: Edge-enhanced hierarchical feature pyramid network for damaged QR code reconstruction”. W: *Mathematics* 11.20 (2023), s. 4349.
- [4] Jinwang Yi i Jianan Chen. „Enhancement of Two-Dimensional Barcode Restoration Based on Recurrent Feature Reasoning and Structural Fusion Attention Mechanism”. W: *Electronics* 13.10 (2024), s. 1873.
- [5] Sungho Suh i in. „Robust Shipping Label Recognition and Validation for Logistics by Using Deep Neural Networks”. W: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, s. 4509–4513. DOI: [10.1109/ICIP.2019.8803412](https://doi.org/10.1109/ICIP.2019.8803412).
- [6] Michał Motyl. „Porównanie metod do wykrywania tekstu na zdjęciach etykiet produktów”. Praca inżynierska. Akademia Górnictwa Hutnicza im. Stanisława Staszica w Krakowie, 2023.
- [7] T. Soni Madhulatha. *An Overview on Clustering Methods*. 2012. arXiv: [1205.1117](https://arxiv.org/abs/1205.1117) [cs.DS].
- [8] Yan Song i in. „A novel image text extraction method based on k-means clustering”. W: *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*. IEEE. 2008, s. 185–190.
- [9] Zespół InPost. *Nowa etykieta przewozowa w formacie ZPL i EPL2 od 8 maja 2023 r.* <https://inpost.pl/aktualnosci-nowa-etynieta-przewozowa-w-formacie-zpl-i-epl2-od-8-maja-2023-r>. Accessed on 06.08.2024. 2023.
- [10] Joseph Redmon i in. „You Only Look Once: Unified, Real-Time Object Detection”. W: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

- [11] Michael Teutsch i Wolfgang Kruger. „Robust and Fast Detection of Moving Vehicles in Aerial Videos Using Sliding Windows”. W: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2015.
  - [12] Ross Girshick i in. „Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. W: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.
  - [13] Mingxing Tan, Ruoming Pang i Quoc V. Le. „EfficientDet: Scalable and Efficient Object Detection”. W: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.
  - [14] Tsung-Yi Lin i in. „Focal Loss for Dense Object Detection”. W: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.
  - [15] Juan Terven, Diana-Margarita Córdova-Esparza i Julio-Alejandro Romero-González. „A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas”. W: *Machine Learning and Knowledge Extraction 5.4* (2023), s. 1680–1716.
  - [16] Mark Everingham i in. „The pascal visual object classes challenge: A retrospective”. W: *International journal of computer vision* 111 (2015), s. 98–136.
  - [17] Alexander Neubeck i Luc Van Gool. „Efficient non-maximum suppression”. W: *18th international conference on pattern recognition (ICPR'06)*. T. 3. IEEE. 2006, s. 850–855.
  - [18] Christian Szegedy i in. „Going Deeper With Convolutions”. W: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.
  - [19] Olga Russakovsky i in. „ImageNet Large Scale Visual Recognition Challenge”. W: *International Journal of Computer Vision (IJCV)* 115.3 (2015), s. 211–252. DOI: [10.1007/s11263-015-0816-y](https://doi.org/10.1007/s11263-015-0816-y).
  - [20] Chien-Yao Wang, I-Hau Yeh i Hong-Yuan Mark Liao. „You only learn one representation: Unified network for multiple tasks”. W: *arXiv preprint arXiv:2105.04206* (2021).
  - [21] Zheng Ge i in. „Yolox: Exceeding yolo series in 2021”. W: *arXiv preprint arXiv:2107.08430* (2021).
  - [22] Glenn Jocher, Ayush Chaurasia i Jing Qiu. *Ultralytics YOLOv8*. Wer. 8.0.0. 2023.
  - [23] Joseph Redmon i Ali Farhadi. „Yolov3: An incremental improvement”. W: *arXiv preprint arXiv:1804.02767* (2018).
  - [24] Glenn Jocher. *YOLOv5 by Ultralytics*. Wer. 7.0. 2020. DOI: [10.5281/zenodo.3908559](https://doi.org/10.5281/zenodo.3908559).
-

- [25] Tsung-Yi Lin i in. „Feature Pyramid Networks for Object Detection”. W: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [26] Chien-Yao Wang i in. „CSPNet: A new backbone that can enhance learning capability of CNN”. W: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, s. 390–391.
- [27] Chien-Yao Wang, Hong-Yuan Mark Liao i I-Hau Yeh. „Designing network design strategies through gradient path analysis”. W: *arXiv preprint arXiv:2211.04800* (2022).
- [28] Boris Epshtein, Eyal Ofek i Yonatan Wexler. „Detecting text in natural scenes with stroke width transform”. W: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 2010, s. 2963–2970. DOI: [10.1109/CVPR.2010.5540041](https://doi.org/10.1109/CVPR.2010.5540041).
- [29] Huizhong Chen i in. „Robust text detection in natural images with edge-enhanced Maximally Stable Extremal Regions”. W: *2011 18th IEEE International Conference on Image Processing*. 2011, s. 2609–2612. DOI: [10.1109/ICIP.2011.6116200](https://doi.org/10.1109/ICIP.2011.6116200).
- [30] Xinyu Zhou i in. „EAST: An Efficient and Accurate Scene Text Detector”. W: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.
- [31] *Tesseract documentation*. <https://tesseract-ocr.github.io/tessdoc/>. 2024.
- [32] Ray Smith. „An overview of the Tesseract OCR engine”. W: *Ninth international conference on document analysis and recognition (ICDAR 2007)*. T. 2. IEEE. 2007, s. 629–633.
- [33] Yong Yu i in. „A review of recurrent neural networks: LSTM cells and network architectures”. W: *Neural computation* 31.7 (2019), s. 1235–1270.
- [34] Pengcheng He i in. *DeBERTa: Decoding-enhanced BERT with Disentangled Attention*. 2021. arXiv: [2006.03654 \[cs.CL\]](https://arxiv.org/abs/2006.03654).
- [35] Jacob Devlin i in. „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. W: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Red. Jill Burstein, Christy Doran i Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, czer. 2019, s. 4171–4186. DOI: [10.18653/v1/N19-1423](https://doi.org/10.18653/v1/N19-1423).

- [36] Shijie Wu i Mark Dredze. „Beto, Bentz, Becas: The Surprising Cross-Lingual Effectiveness of BERT”. W: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Red. Kentaro Inui i in. Hong Kong, China: Association for Computational Linguistics, list. 2019, s. 833–844. DOI: [10.18653/v1/D19-1077](https://doi.org/10.18653/v1/D19-1077).
- [37] Muhammad Hussain. „YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection”. W: *Machines* 11.7 (2023), s. 677.
- [38] Alexander Mordvintsev i K Abid. „Opencv-python tutorials documentation”. W: *Obtenido de https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf* (2014).
- [39] CVAT.ai Corporation. *Computer Vision Annotation Tool (CVAT)*. Wer. 2.8.2. List. 2023.
- [40] J. D. Hunter. „Matplotlib: A 2D graphics environment”. W: *Computing in Science & Engineering* 9.3 (2007), s. 90–95. DOI: [10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55).
- [41] Michael L. Waskom. „seaborn: statistical data visualization”. W: *Journal of Open Source Software* 6.60 (2021), s. 3021. DOI: [10.21105/joss.03021](https://doi.org/10.21105/joss.03021).
- [42] Charles R. Harris i in. „Array programming with NumPy”. W: *Nature* 585.7825 (wrz. 2020), s. 357–362. DOI: [10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2).
- [43] Wes McKinney i in. „Data structures for statistical computing in python”. W: *Proceedings of the 9th Python in Science Conference*. T. 445. Austin, TX. 2010, s. 51–56.
- [44] Matthias A Lee i in. *Pytesseract*. <https://tesseract-ocr.github.io/tessdoc/>. 2017.
- [45] Andrew Cameron Morris, Viktoria Maier i Phil D Green. „From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition.” W: *Inter-speech*. 2004, s. 2765–2768.
- [46] Xiangyang Xu i in. „Characteristic analysis of Otsu threshold and its applications”. W: *Pattern recognition letters* 32.7 (2011), s. 956–961.