

## **Streszczenie**

Niniejsza praca dyplomowa podejmuje problematykę identyfikacji ludzi z stop-klatek pochodzących z kamer przemysłowych z wykorzystaniem głębokich sieci neuronowych. W pracy zarówno zbadano dostępne architektury typu Res-Net, jak i zaproponowano użycie własnej modyfikacji jakim jest wykorzystanie architektury typu EfficientNet, którą dodano w ramach framework-u reid-strong-baseline. Wykonano szereg testów zarówno na ogólnie dostępnym zbiorze danych Market1501 jak i stworzonych autorskich zbiorów do celów ewaluacyjnych.

## **Abstract**

This paper contains research of re identification problem of people in set of images generated from industrial cameras video frames. The solution was based on deep neural networks (GSN). The work both examined the available Res-Net architectures and proposed the use of its own modification, which is the use of the EfficientNet architecture, which was added as part of the reid-strong-baseline framework. A number of tests were performed on both the general available Market1501 data set and the created own data sets for evaluation purposes.

## Spis treści

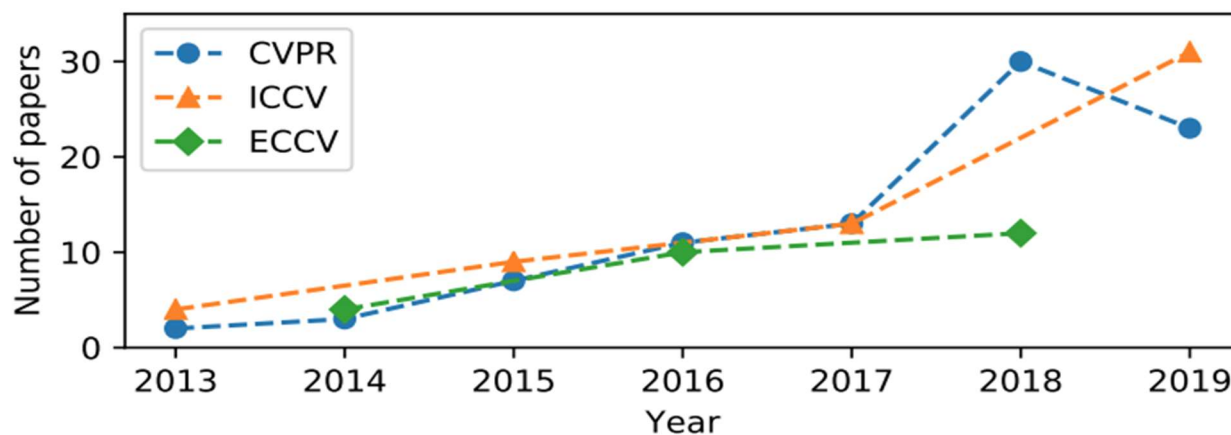
Streszczenie.....	1
Abstract .....	1
1. Wstęp .....	4
1.1. Motywacja .....	4
1.2. Definicja problemu .....	5
1.3. Cel pracy.....	6
2. Wybór systemów do reidentyfikacji .....	6
3. Zbiory danych (Data sets) i wybrane metryki.....	8
3.1. Market1501 .....	10
3.2. Duke MTMC .....	11
3.3. Własny zbiór danych .....	11
3.4. Zbiór wygenerowany z nagrania pobranego z YouTube .....	12
3.5. Podsumowanie wyboru dataset-u .....	13
3.6. Augmentacja danych .....	13
3.7. Metryki oceny modeli.....	15
3.8. Mean Avarage Precision (mAP) .....	15
3.9. Czas inferencji modelu .....	17
3.10. Miara Cumulative Match Characteristic (CMC) .....	17
4. Modele .....	18
4.1. Model Resnet50 .....	18
4.2. EfficientNet w wersji b0 .....	19
4.3. Porównanie modeli pod względem rozmiaru i ilości parametrów .....	20
5. Rozwiązania optymalizujące proces trenowania .....	21
5.1. Rozgrzewanie kroku uczenia (Warmup Learning Rate).....	21
5.2. Losowo wycinanie (Random Erasing Augmentation).....	22

5.3.	Wygładzanie adnotacji (Label Smoothing) .....	22
5.4.	Zmiana ostatniej warstwy konwolucyjnej (Last Stride) .....	23
5.5.	BBNeck.....	24
5.6.	Center Loss .....	25
5.7.	Podsumowanie .....	26
6.	Rozwiązanie bazowe.....	26
6.1.	Własna ewaluacja .....	28
7.	Modyfikacje .....	30
7.1.	Użycie jako sieci bazowej sieci EfficientNet .....	30
7.2.	Zmniejszenie rozmiaru embeddingu.....	30
7.3.	Zmniejszenie precyzji zapisywanych parametrów do float16.....	30
7.4.	Wprowadzenie dodatkowej augmentacji danych .....	31
7.5.	Wprowadzenie funkcji Swish jako funkcji aktywacji .....	31
8.	Trening zmodyfikowanych modeli .....	32
9.	Wnioski .....	34
9.1.	Wnioski na podstawie mAP.....	34
9.2.	Wnioski na podstawie czasu inferencji .....	35
10.	Podsumowanie.....	36
10.1.	Komentarz uzyskanych wyników .....	36
10.2.	Dalsze badania .....	36
11.	Bibliografia.....	38
12.	Spis rysunków .....	39

# 1. Wstęp

## 1.1. Motywacja

Problem rozpoznawania osób i obiektów na różnych ujęciach z wielu kamer stał się w ostatnim czasie jednym z najczęściej badanych zagadnień. Potwierdza to cytat z pracy [4] *"Driven by the growing demands for intelligent surveillance and forensic applications, person re-identification (re-ID) has become a topical research area in computer vision."* w wolnym tłumaczeniu "Rozwiązanie umożliwiające identyfikację osób z różnych kamer oraz ujęć stało się jednym z najczęściej badanych zagadnień. Stało się tak za sprawą rosnącego zainteresowania służb wykorzystaniem aplikacyjnym takiego rozwiązania". Na potwierdzenie tego wniosku przedstawiono wykres obrazujący liczbę publikacji traktujących o tej tematyce na przestrzeni lat.



Rys 1 Zestawienie ilości publikacji dotyczących reidentyfikacji w latach.[6]

Potencjał rozwiązania tego zagadnienia wykracza jednak poza użycie jakim są zainteresowane służby bezpieczeństwa. Jest zdecydowanie bardziej ogólnym problemem dającym możliwość przypisywania identyfikatora dla obiektów tej samej klasy. Jest zatem kontynuacją procesu detekcji. Rozszerza również dziedzinę rozwiązań o klasy pośrednie w stosunku do wykorzystanych w procesie uczenia.

Warto również zaznaczyć, że zgodnie z [6] znaczący postęp w tej tematyce dokonał się właśnie dzięki GSN (głębokim sieciom neuronowym). Odnosząc się do cytatu [6] *"Person re-identification (ReID) with deep neural networks has made progress and achieved high performance in recent years. However, many state-of-the-arts methods design complex network structure and concatenate*

*multibranch features. In the literature, some effective training tricks or refinements are briefly appeared in several papers or source codes*" należy zauważyć, że zgodnie z przytoczonym fragmentem rozwój w tej dziedzinie jest bardzo dynamiczny i istnieje przestrzeń na łączenie wielu proponowanych rozwiązań. Ponad to istnieje szerokie pole do usprawnień i proponowania własnych strategii rozwiązań tego istotnego problemu.

## **1.2. Definicja problemu**

Kluczowym krokiem do rozwiązania jakiegokolwiek problemu jest jego uprzednie poprawne zdefiniowanie. Z tego powodu na kolejnych liniach tekstu sprecyzowane zostanie pojęcie reidentyfikacji, które będzie w tej pracy używane wymiennie z pojęciem identyfikacji.

W tej pracy problem reidentyfikacji będzie odnosił się do reidentyfikacji osób na stop klatkach z wielu kamer lub jednej kamery w różnych chwilach czasowych. Oznacza to dopasowywanie obrazów ze zbioru danych, przedstawiających tych samych ludzi i ich grupowanie. Zbiór obrazów charakteryzuje się ujęciami o słabej rozdzielczości osób wykonanymi w różnym oświetleniu, z wielu odmiennych perspektyw. Z tego powodu założono, że ta sama osoba inaczej ubrana jest w tym przypadku traktowana jako inny człowiek.

Próbę uogólnienia definicji reidentyfikacji zaproponowaną przez autora umieszczono poniżej. Reidentyfikacja obiektu to proces generowania zbioru informacji o obiekcie opisującym go niezależnie od zmiennych arbitralnie uznanych za nieistotne oraz cech narzuconych przez sprzęt przetwarzający rzeczywistość na dane digitalowe.

Obiekt rozumiany jest tu jako dowolna rzecz materialna mająca swoje odzwierciedlenie w rzeczywistości.

Definicja zawiera więc arbitralną granicę tego co uważamy za ten sam obiekt. Przykładem może być tu człowiek identyfikowany z imienia i nazwiska, który przy założeniu ubioru jako nieistotnej cechy obiektu, niezależnie od niego zostanie uznany za ten sam obiekt. W tym opracowaniu, jednak sytuacja w której ta sama osoba zostaje uchwycona przez kamerę w innym ubraniu traktowana jest jako inny obiekt. Natomiast cechy wynikające z położenia osoby względem kamery czy oświetlenia sceny traktowane są jako cechy, które powinny zostać pominięte w procesie tworzenia opisu obiektu.

Istnieje wiele sposobów na realizację pojęcia „opisu” obiektu, w dalszej części pracy będzie to jednak utożsamiane z wektorem wartości o wynikającej z specyfiki modelu wymiarowości.

### 1.3. Cel pracy

Celem pracy jest sprawdzenie jak pretrenowana na zbiorze ImageNet sieć EfficientNet jest skutecznym narzędziem do wyodrębniania cech. Cechy te w dalszej toku przetwarzania wykorzystywane są do generowania wektora opisującego obiekt (embeddingu). Sieć EfficientNet zostanie porównana z obecnie osiągnającą najlepsze wyniki siecią ResNet50. Poza kryterium jakości tworzonego opisu obiektu, w tym kontekście rozumianym jako Mean Average Precision (mAP) sprawdzony zostanie czas interferencji, czyli czas przetwarzania.

Powodem, dla którego istotny jest czas przetwarzania poza jakością generowanych wektorów cech głębokich (embeddingów) jest to, że proponowane rozwiązanie potencjalnie mogłoby znaleźć zastosowanie w roli systemu wspierającego śledzenie obiektów na nagraniach. W takim rozwiązaniu, pracującym w czasie rzeczywistym, ilość koniecznych do wykonania operacji procesora w celu wygenerowania opisu obiektu jest kluczowa do zapewnienia sensowności użycia tego podejścia. Zbyt długi czas przetwarzania wyeliminuje możliwość pracy w trybie rzeczywistym co jest priorytetową potrzebą biznesową.

## 2. Wybór systemów do reidentyfikacji

Ze względu na popularność oraz to jak istotne we współczesnych badaniach jest zagadnienie reidentyfikacji, zostało już stworzonych oraz opublikowanych wiele systemów (frameworki) zawierających środowisko do rozwijania i testowania tego zagadnienia. W celach porównawczych zdecydowano o wykorzystaniu jednego z nich. Zapewnia to porównywalność wyników oraz przyspiesza pracę nad autorskimi modyfikacjami. W ocenie autora dwa najbardziej popularne frameworki to:

- [1] *reid-strong-baseline*

oraz

- [5] *Torchreid: Library for Deep Learning Person Re\_Identification in Pytorch*

Oba systemy oferują bogatą pulę narzędzi do wykorzystania wielu zbiorów danych, odmiennych trybów uczenia oraz pozostawiają przestrzeń do własnych modyfikacji. Zawierają implementacje technik augmentacji danych, doboru przykładów do uczenia (hard sampling) oraz co istotne różnych sieci bazowych (back bone model).

Po analizie obu systemów zdecydowano o wykorzystaniu [1] *reid-strong-baseline*. O wyniku zadecydowała bogata dokumentacja otrzymanych wyników uzyskanych przy użyciu framework-u, z rozbiciem na wykorzystywane techniki optymalizacji procesu trenowania. Istotnym aspektem było również to, że obecne najlepsze rozwiązanie zostało opublikowane właśnie przy użyciu tego systemu przez jego autorów [6]. Jest to framework napisany z użyciem PyTorch i jest kompleksowym narzędziem do tworzenia i porównywania sieci stworzonych do reidentyfikacji. Zawiera wiele baz danych.

Poniżej zaprezentowano wyniki jakie zawarto w repozytorium github framework-u *reid-strong-baseline*:

Results (rank1/mAP)		
Model	Market1501	DukeMTMC-reID
Standard baseline	87.7 (74.0)	79.7 (63.8)
+Warmup	88.7 (75.2)	80.6(65.1)
+Random erasing augmentation	91.3 (79.3)	81.5 (68.3)
+Label smoothing	91.4 (80.3)	82.4 (69.3)
+Last stride=1	92.0 (81.7)	82.6 (70.6)
+BNNeck	94.1 (85.7)	86.2 (75.9)
+Center loss	94.5 (85.9)	86.4 (76.4)
+Reranking	95.4 (94.2)	90.3 (89.1)

Backbone	Market1501	DukeMTMC-reID
ResNet18	91.7 (77.8)	82.5 (68.8)
ResNet34	92.7 (82.7)	86.4(73.6)
ResNet50	94.5 (85.9)	86.4 (76.4)
ResNet101	94.5 (87.1)	87.6 (77.6)
ResNet152	80.9 (59.0)	87.5 (78.0)
SeResNet50	94.4 (86.3)	86.4 (76.5)
SeResNet101	94.6 (87.3)	87.5 (78.0)
SeResNeXt50	94.9 (87.6)	88.0 (78.3)
SeResNeXt101	95.0 (88.0)	88.4 (79.0)
IBN-Net50-a	95.0 (88.2)	90.1 (79.1)

Rys 2 : Zestawienie wyników uzyskanych w pracy [6]

Wskazują one na wyniki jakie otrzymano wykorzystując każdą z kolejnych modyfikacji treningów jakie opisane zostaną w punkcie 6 tej pracy. Rozróżniono w niej wyniki wykonane na podstawie dwóch różnych datasetów:

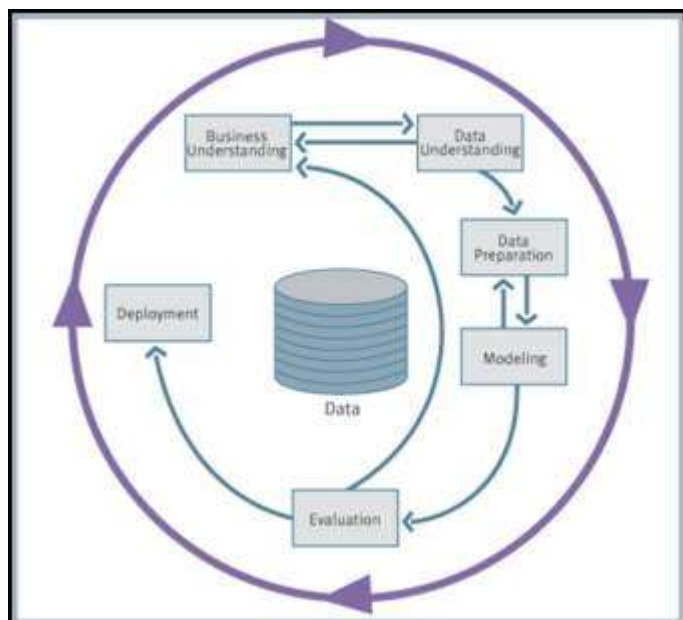
- Market1501
- DukeMTMC-reID

Pozwala to na estymację jak każda ze zmian w procesie trenowania może poprawić wyniki dla nowo trenowanej sieci.

Wyniki odnoszące się do różnych sieci bazowych kreują obraz w którym najlepszą siecią bazową wykorzystaną przez autorów systemu ***reid-strong-baseline*** jest IBN-Net50-a, a najgorszą ResNet156. Średni wynik oscyluje w granicach 85% dla mAP, który pokrywa się z wynikiem sieci ResNet50, którą przyjęto jako reprezentatywną.

### 3. Zbiory danych (Data sets) i wybrane metryki.

Zgodnie z zasadami CRISP DM wyznaczającymi najlepsze praktyki w tworzeniu rozwiązań AI, dane uczące zajmują, ważne miejsce w procesie tworzenia systemu. To jakość danych determinuje ostateczną użyteczność tworzonego rozwiązania, a systemy oparte o uczenie na danych są tak dobre, jak dobry jest zbiór z danymi.



Rys 3 Cykl tworzenia systemów data mining CRISP DM

<https://media2.picsearch.com/is?0y5BVmUtw6Ychs3zIkqkKxe79hNRgDbshMID-namuPY&height=309x>



Ponieważ liczba zbiorów danych uczących jest tak istotna, opracowano zestaw cech jakie powinien charakteryzować zbiory danych. Istotne w zagadnieniu rozpatrywanym w tej pracy jest zapewnienie, aby zbiór danych wykorzystywany do trenowania modeli tworzących embeddingi był:

- 1) Poprawny – nie zawierał, źle zidentyfikowanych obiektów
- 2) Różnorodny – zawierał wiele różnych obiektów, pokrywających możliwie najszerszą gamę możliwych sytuacji.
- 3) Kompletny – zidentyfikowane obiekty muszą zawierać więcej niż jeden obraz, tak by było możliwe wykorzystanie go w procesie uczenia.
- 4) Zrównoważony – ilość obrazów dla każdego z zidentyfikowanych obiektów powinna być możliwie równa.

Zapewnienie tych reguł stanowić będzie o jakości ostatecznego rozwiązania. Niemniej jednak przygotowanie danych jest również najbardziej pracochłonną część procesu tworzenia rozwiązań opartych o uczenie maszynowe. Zapewnienie prawidłowej jakości danych i pokrycie nim problemu jaki w zamierzeniach ma zostać rozwiązany stanowi wyzwanie.

Poniżej zaprezentowano zaimplementowane do użycia zbiory danych w systemie *reid-strong-baseline*:

Zbiory danych zawierające pojedyncze ujęcia obiektów	Zbiory danych zawierające przetworzone nagrania
Market1501 [7] Market1501 - pdf	MARS
CUHK03	
DukeMTMC-reID	iLIDS-VID
MSMT17	
VIPeR	PRID2011
GRID	
CUHK01	DukeMTMC-VideoReID
SenseReID	

QMUL-iLIDS	
PRID	

Dwa najczęściej wykorzystywane w tego typu zadaniach datasety to:

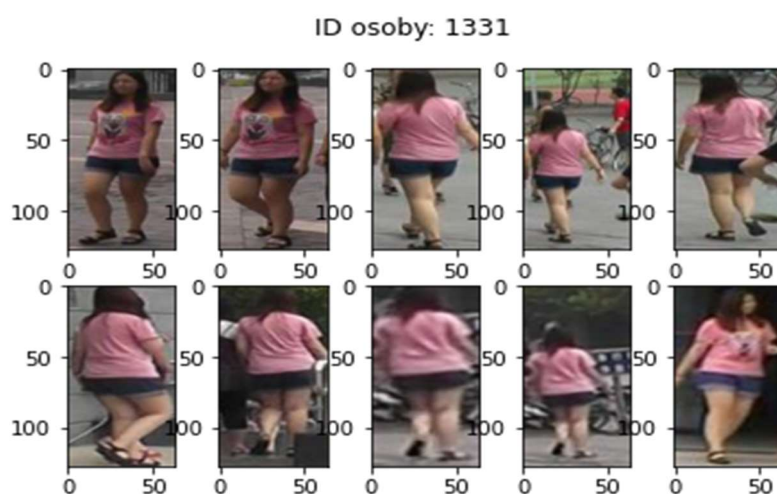
1. Market1501 [7] Market1501 - pdf
2. DukeMTMC-reID

Również dla tych dwóch zbiorów danych zostały podane wyniki procesu trenowania i ewaluacji w pracy [6].

### 3.1. Market1501

To jeden z najczęściej wykorzystywanych zbiór danych do trenowania systemów reidentyfikacji osób. Jak twierdzą twórcy [7] zbiór utrzymuje wysoką jakość danych oraz zawiera szeroką pulę przypadków. Złożony jest z ponad 500 000 obrazów z czego wyodrębniono 32 000 różnych osób zgrupowanych pod jednym identyfikatorem id. Każda z osób posiada wiele ujęć uchwyconych z różnych ujęć i różnych kamer. Wygenerowane obrazy zawierają jedynie jedną osobę oraz ich jakość jest porównywalna dla każdej z osób. Kod przetwarzający strukturę w jakie zapisano dane został zaimplementowany w wielu frameworkach w tym reid-strong-baseline

Przykład ze zbioru Markets1501 zaprezentowano poniżej:



Rys 4 Przykład danych ze zbioru Markets1501

### 3.2. Duke MTMC

To również bardzo popularny w tych zastosowaniach zbiór danych. Został stworzony na kampusie uniwersyteckim w roku 2014 i od tej pory wykorzystywany jest w zagadnieniach reidentyfikacji osób oraz rozpoznawania twarzy w obrazach o niskiej rozdzielczości. Do wygenerowania zbioru wykorzystano 14 godzin nagrań z 8 zainstalowanych kamer. Otrzymano dzięki temu obrazy 2 tysięcy unikatowych osób. Poszerzony opis można znaleźć w [10]

### 3.3. Własny zbiór danych

W celu weryfikacji przydatności modeli do wykorzystania w zadaniu śledzenia osób na nagraniach wideo, stworzono własny zbiór danych. Zbór ten pełnił rolę zbioru porównawczego do oceny jakości tworzenia embeddingów z nagrań kamery 360. Posłużono się modelem kamery powszechnie stosowanym w punktach sprzedaży lokalizowanych w centrach handlowych oraz samodzielnych salonach obsługi klientów.

Zbiór wygenerowano z nagrania



Rys 5 Przykład ujęcia z autorskiego nagrania kamerą 360

Link do nagrania [https://drive.google.com/file/d/1-0vgAB7ujrl-55ZU-8IpkaS\\_7wqbt9Mq/view](https://drive.google.com/file/d/1-0vgAB7ujrl-55ZU-8IpkaS_7wqbt9Mq/view)

Do wyodrębnienia obiektów na nagraniu wykorzystano framework YOLO5 [11] z obiektów wykstrahowanych z nagrania wybrano jedną postać ludzką. Z nagrania o długości **1min26s**

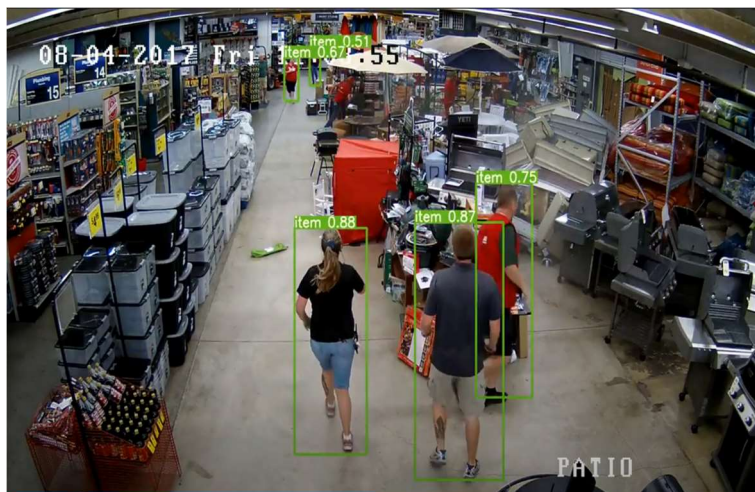
uzyskano **1203** wycięte fragmenty z tą samą postacią w różnych pozach. Przykładowe wycięte obrazy:



*Rys 6 Przykłady wyfiltrowanych obiektów z nagrania kamerą 360*

### **3.4. Zbiór wygenerowany z nagrania pobranego z YouTube**

Zbiór wygenerowano z nagrania umieszczonego poniżej.



*Rys 7 Przykład ujęcia z nagrania pobranego z portalu YouTube*

Link do nagrania: <https://drive.google.com/file/d/1P2AdTCr0f2htDGtW2rmq5qMQ1L3uf3fG/view>

Korzystając z frameworku YOLO5 zmodyfikowanym na potrzeby tej pracy, jedynie do detekcji postaci ludzkich, wygenerowano dataset z wyciętymi osobami z nagrania. Posłużą one do stworzenia embeddingów i wyszukania tej samej osoby z kolejnych klatek nagrania. Stanowi to formę ewaluacji modeli pod względem gotowości użycia w problemie śledzenia osób na stop-klatkach nagrania. Danym nie przypisano identyfikatora id z tego powodu nie możliwe jest porównanie wyników w sposób ścisły.

### **3.5. Podsumowanie wyboru dataset-u**

W tej pracy wykorzystany zostanie jedynie zewnętrzny dataset Market1501 oraz własne datasey. Jeden z nich posłuży do określenia czasu interferencji drugi natomiast poprawności odnajdywania tej samej osoby w zbiorze obrazów wygenerowanych z wideo. Zdecydowano o użyciu jednego zbioru danych z powodu ograniczeń sprzętowych oraz by w procesie porównawczym wyeliminować wpływ doboru danych uczących, które przy wielu zbiorach danych w połączeniu z ograniczoną liczbą epok uczenia znacząco wpływałyby na ostateczny wynik modeli. Szczegółowy opis zbioru danych znajduje się w pracy [7]. Analiz zbioru danych zawarta jest w notebooku

[https://github.com/tomektarabasz/Praca\\_Dyplomowa\\_Tomasz\\_Tarabasz/blob/master/notebooks/PD\\_data\\_sets.ipynb](https://github.com/tomektarabasz/Praca_Dyplomowa_Tomasz_Tarabasz/blob/master/notebooks/PD_data_sets.ipynb)

Własny dataset zostanie wykorzystany jedynie w celu wyciągnięcia wniosków o jakości modeli w dwóch wybranych kryteriach:

- zbieżności embeddinów dla dataset-u złożonego z jednej postaci
- czasu przetwarzania.

Parameter czasu przetwarzania jest szczególnie istotny pod względem wykorzystania w systemie śledzenia i identyfikacji rozpoznanych sylwetek ludzkich


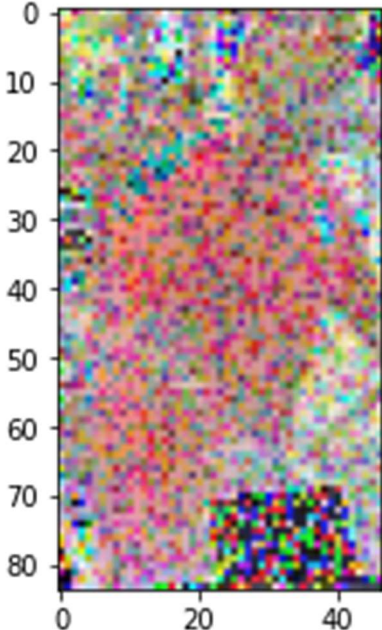
### **3.6. Augmentacja danych**

To zbiór technik pozwalających na sztuczne rozszerzenie dostępnego zbioru danych. Pozwalają na wytworzenie sztucznych obrazów na podstawie już istniejących. Pozwala to „nakierowanie” treningu modelu na cechy jakie byłyby trudne do wytrenowania na „naturalnym” zbiorze danych.

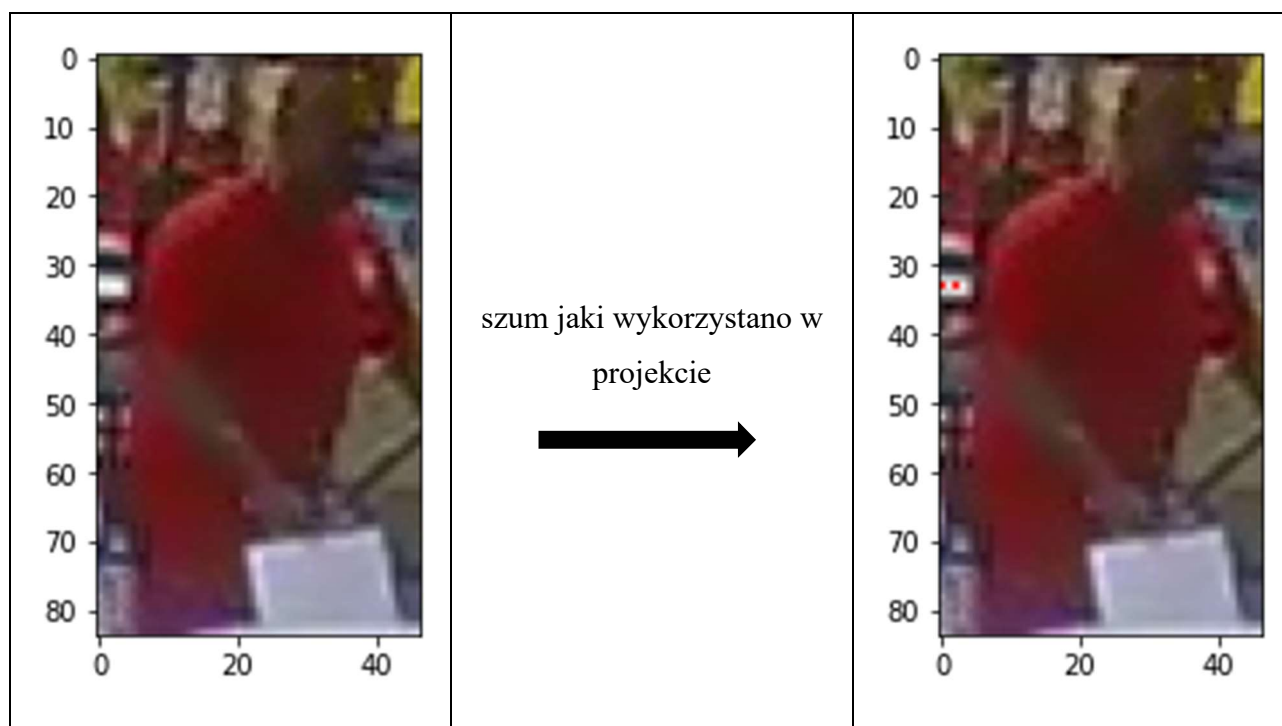
## Własna augmentacja

Generalizacja modelu jest trudna do osiągnięcia. Modele zazwyczaj są tak „zgeneralizowane” jak obszerny jest zbiór danych uczących. Możliwość wykorzystania modeli jest jednak uzależniona od tego w jak szerokim zakresie przypadków daje on spodziewane wyniki. Walcząc więc o większą generalizację zaproponowano dodatkową augmentację.

Ideą jest uniezależnienie embeddingów od nieistotnych prawidłowości w obrazach z datasetu. Przypadkowy szum dodany na niskim poziomie luminancji ma pomagać w generalizacji ostatecznie wytrenowanej sieci.

Oryginalny	➔	Przetworzony
	<p>szum 50x w stosunku do użytego w projekcie (cel prezentacyjny)</p> <p>➔</p>	





Inspiracją do wykorzystania przypadkowego szumu jest technika ataku na modele nazywana „adversarial attack”. Polega na dodaniu uprzednio wytrenowanej maski do obrazu wejściowego, co ostatecznie „podbija” wagi w modelu, odpowiadające cechom klasy na którą trenowana była maska. W założeniach ataku, „myli” to model zmniejszając jego precyzję lub w skrajnym przypadku generującym błędną odpowiedź modelu.

### 3.7. Metryki oceny modeli

Metryka wykorzystana do trenowania modelu jest kluczowym parametrem

Zdecydowano by oceniać modele pod względem dwóch podstawowych metryk:

- Mean Avarage Precision (mAP)
- Czas inferencji modelu

### 3.8. Mean Avarage Precision (mAP)

W celach porównawczych wytrenowanych modeli, zdecydowano o wykorzystania miary jaką jest Mean Avarage Preciosion (mAP). Daje ona wiarygodną informację o jakości modelu względem sieci bazowej. Jako sieć bazową przyjęto najlepszą sieć z podaną w publikacji [6]. Wyniki dla innych sieci testowanych przez autorów przytoczonego artykułu są również podane w artykule i odnoszą się do

tej właśnie miary. Dzięki temu możliwe było odniesienie się uzyskanymi wynikami do wielu przykładów przebadanych w publikacji [6]

Mean Average Precision wyliczane jest z użyciem pojęć takich jak:

- Zbieżność cosinusowa wektorów – odnosząc się do zadań detekcji, zastępuje pojęcie IoU (intersection over union), ta miara określała czy wyznaczona detekcja faktycznie odnosi się do istniejącego obiektu poprzez porównanie części pola wspólnego między ramką odniesienia (ground truth), a wyznaczoną ramką. Arbitralna wartość 0,5 stanowiła granicę czy detekcja była prawdziwa czy nie. W przypadkach reidentyfikacji tę rolę pełni porównanie zbieżności wektorów embeddingów. Obraz pytający (query image) porównywany jest ze zbiorem wektorów wyznaczonych dla wszystkich obrazów. Poziom po którym uznaje się, że obraz został poprawnie zakwalifikowany to przypadek w, którym znaleziono odpowiadający wektor ze zbieżnością cosinusową większą niż 0.7
- Precyzja (precision) – definiowana jest jako proporcja między prawidłowo wskazanymi obrazami, a wszystkimi przypadkami, gdzie przypisano obraz (na podstawie zbieżności cosinusowej):

$$\text{Precyzja} = \frac{TP}{TP + FP}$$

gdzie:

*TP – to prawidłowa identyfikacja*

*FP – to fałszywa identyfikacja*

- Czułość (recall) – definiowane jako proporcja między prawidłowo wskazanymi obrazami, a wszystkimi możliwymi detekcjami. W tym przypadku odnosi się to do proporcji odnalezionych, w kontekście przyjętej zbieżności cosinusowej, obrazów danej osoby z pośród wszystkich obrazów danej osoby.

$$\text{Czułość} = \frac{TP}{TP + FN}$$

gdzie:

*TP – to prawidłowa identyfikacja*

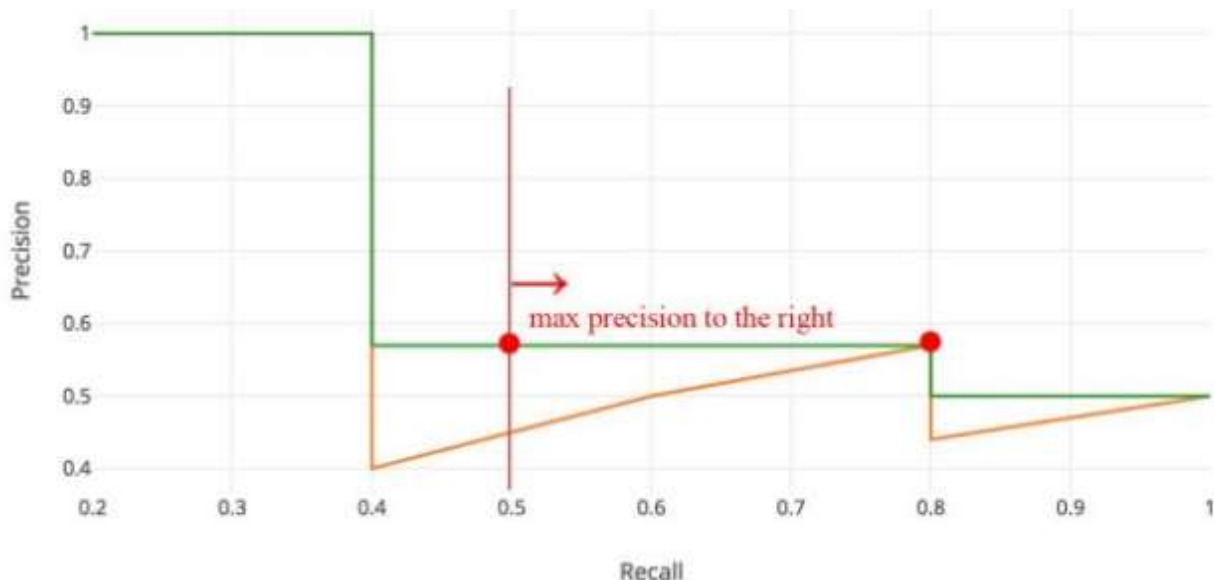
*FN – to fałszywa nieidentyfikacja*

- Average Precision (AP) - jest polem pod krzywą wyznaczoną z przedstawienia wartości czułości i precyzji na jednym wykresie. Opisuje to formuła:



$$AP = \int_0^1 \text{precyzja}(\text{czułość}) d \text{czułość}$$

Mean Average Precision wyznaczana jest poprzez wyznaczenie AP dla nowej krzywej, wyznaczonej jako krzywa schodkowa o wartościach równych maksymalnej wartości występujących przy rosnących wartościach czułości. Najlepiej obrazuje to grafika umieszczona poniżej:



Rys 8 Krzywa do wyznaczani mAP [https://medium.com/@jonathan\\_hui/map-mean-average-precision-for-object-detection-45c121a31173](https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173)

### 3.9. Czas inferencji modelu

Ponieważ jednym z celów jakie wyznaczono dla tej pracy było przetestowanie możliwości użycia tego typu rozwiązania do śledzenia postaci na nagraniach w czasie rzeczywistym, ważnym aspektem oprócz jego precyzji jest również czas przetwarzania.

Czas przetwarzania przeliczono na podstawie własnego datasetu liczącego około 1200 obrazów. Podano wyniki dla testowanych sieci oraz zestawiono je tabelarycznie.

### 3.10. Miara Cumulative Match Characteristic (CMC)

Jest to miara pokazująca czy obraz o klasie „x” na podstawie swojego embeddingu został zaklasyfikowany jako klasa „x” lub czy występuje w zbiorze (1,5,10) najbliższych wyników. Do określenia „bliskości” można wykorzystać różne miary podobieństwa wektorów. Jest to dystans między wektorami lub podobieństwo cosinusowe. Miara CMC będzie podawana w testach sieci wykorzystywanych w tej pracy.

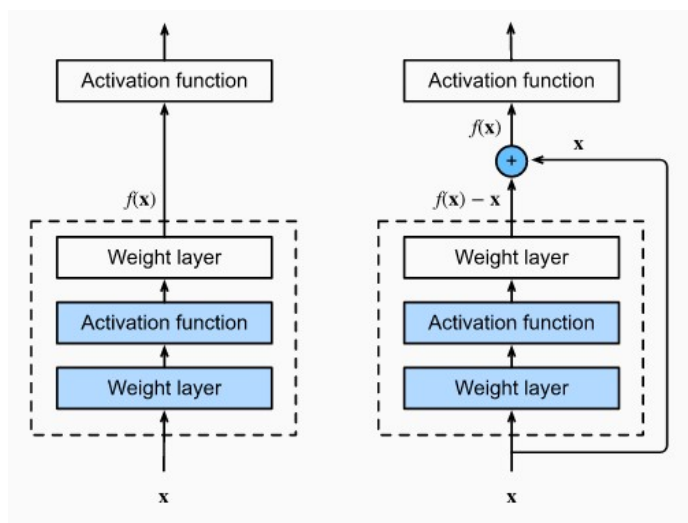
## 4. Modele

Zgodnie z artykułem [6] modelem osiągającym najlepsze wyniki jest model Resnet50 wytrenowany na ImageNet. Jego opis umieszczony zostanie poniżej. Jego wyniki będą poziomem referencyjnym.

Ponieważ zgodnie z [21] model EfficientNet jest modelem znacznie szybszym od Resnet50 oraz faktu, że na chwilę pisania tej pracy EfficientNet osiągnął najwyższy wynik zadaniu klasyfikacji obrazów na zbiorze ImageNet, zdecydowano o próbie zastąpienia modelu Resnet50 modelem EfficientNet-b0.

### 4.1. Model Resnet50

Rodzina modeli ResNet wyróżnia się użyciem charakterystycznego bloku rezydualnego. Rozwiązanie to było rewolucyjnym pomysłem pomagającym w powszechnie występującym problemie „zanikającego” gradientu. Wprost ze wzrostem głębokości sieci neuronowej wzrasta ryzyko bezproduktywnej propagacji wstecznej, która nie jest w stanie modyfikować parametrów głębokich warstw modelu, w tym kontekście oznaczającym warstwy najbardziej „oddalone” od funkcji strat. Przykład bloku rezydualnego zaprezentowano poniżej:



Rys 9 Przykład bloku rezydualnego [http://d2l.ai/chapter\\_convolutional-modern/resnet.html](http://d2l.ai/chapter_convolutional-modern/resnet.html)

Modyfikacja polegająca na sumowaniu wyników przetwarzania warstwy rezydualnej z jej warstwą ją poprzedzającą pozwala na efektywniejszą propagację gradientu. Model ResNet50 zawdzięcza numer 50 ilości warstw rezydualnych wykorzystanych do stworzenia sieci. Wykorzystana sieć została uprzednio wytrenowana na zbiorze ImageNet, a jej wagi zaciągnięte do zainicjowanego modelu.

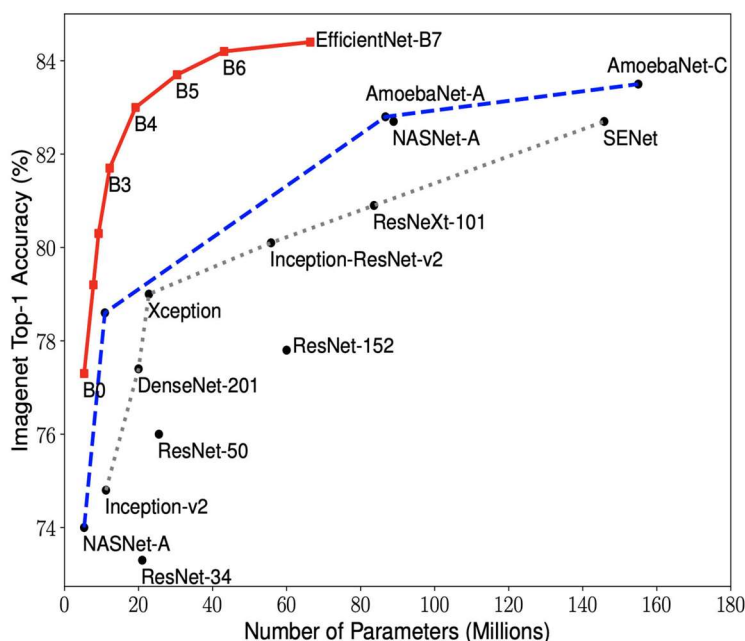
## 4.2. EfficientNet w wersji b0

Repozytorium z implementacją sieci EfficientNet z użyciem biblioteki PyTorch umieszczono w [2].

Repozytorium zawiera wytrenowane modele na bazie obrazów ImageNet. Posiada również zaimplementowane metody do ewaluacji wytrenowanej sieci na własnym zbiorze danych.

Zgodnie z opisem zawartym w repozytorium EfficientNet [12](<https://github.com/lukemelas/EfficientNet-pyTorch?fbclid=IwAR28bdEvf05yCsMdE7ByUP5z-6EyRmadPp5EoyLd57nahLfsikDEuiIT7eU>) to model należący do grupy modeli dedykowanych klasyfikacji obrazów. Uzyskał on, na moment pisanie tej pracy, najlepsze wyniki w tej grupie zadań. Ponad to jest mniejszym i szybszym modelem niż jego poprzednicy. Jest wzorowany na modelach AutoML oraz Compound Scaling.

W szczególności model EfficientNet-B0 jest modelem o rozmiarze odpowiadającym rozwiązaniom mobilnym, który został stworzony jako rozwinięcie AutoML Mobile framework. Poniżej zaprezentowano wykresy prezentujące liczbę parametrów dla kilku z wiodących rozwiązań w porównaniu do EfficientNet



Rys 10 Zestawienie różnych sieci neuronowych na podstawie ilości parametrów oraz wyniku na zbiorze danych ImageNet

Autorzy rozwiązania wymieniają osiągnięcia modelu jako:

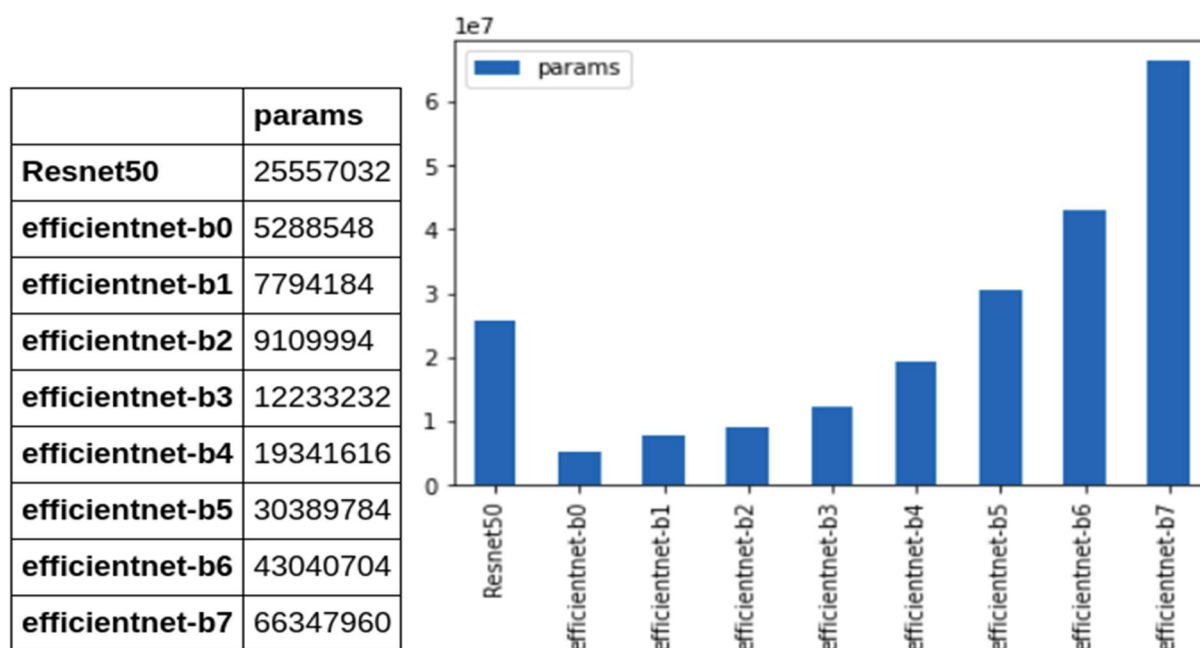
Przy wysokim poziomie dokładności wyników sieć EfficientNet-B7 osiągnęło najlepszy wynik o poziomie 84.4% dla "top-1" oraz 97.1% wśród pięciu najlepszych wyników ("top-5") na zbiorze danych ImageNet z 66 milionami parametrów i 37B FLOPS. Jest to model 8.4 razy mniejszy i 6.1 razy szybszy na CPU od swojego poprzednika Gpipe ([8] <https://arxiv.org/pdf/1811.06965.pdf>).

Przy średnim poziomie dokładności wyników, model EfficientNet-B1 jest 7,6 razy mniejszy i 5,7 razy szybszy na CPU od ResNet-152 ([9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun: "Deep Residual Learning for Image Recognition", 2015) z porównywanym wynikiem dokładności dla ImageNet.

W porównaniu z powszechnie używanym ResNet-50 model EfficientNet-B4 poprawia wyniki dla "top-1" rezultatów, czyli najbardziej prawdopodobnej klasyfikacji, o 6.3% (z poziomu 76,3% do 82.6%) dla tego samego poziomu FLOPS świadczącym o szybkości uzyskania predykcji.

Są to powody dla których zdecydowano o próbie wykorzystania tej architektury jako "BackBone", czyli ekstraktora cech w problemie reidentyfikacji osób.

#### 4.3. Porównanie modeli pod względem rozmiaru i ilości parametrów



Rys 11 Zestawienie modeli pod względem ilości parametrów

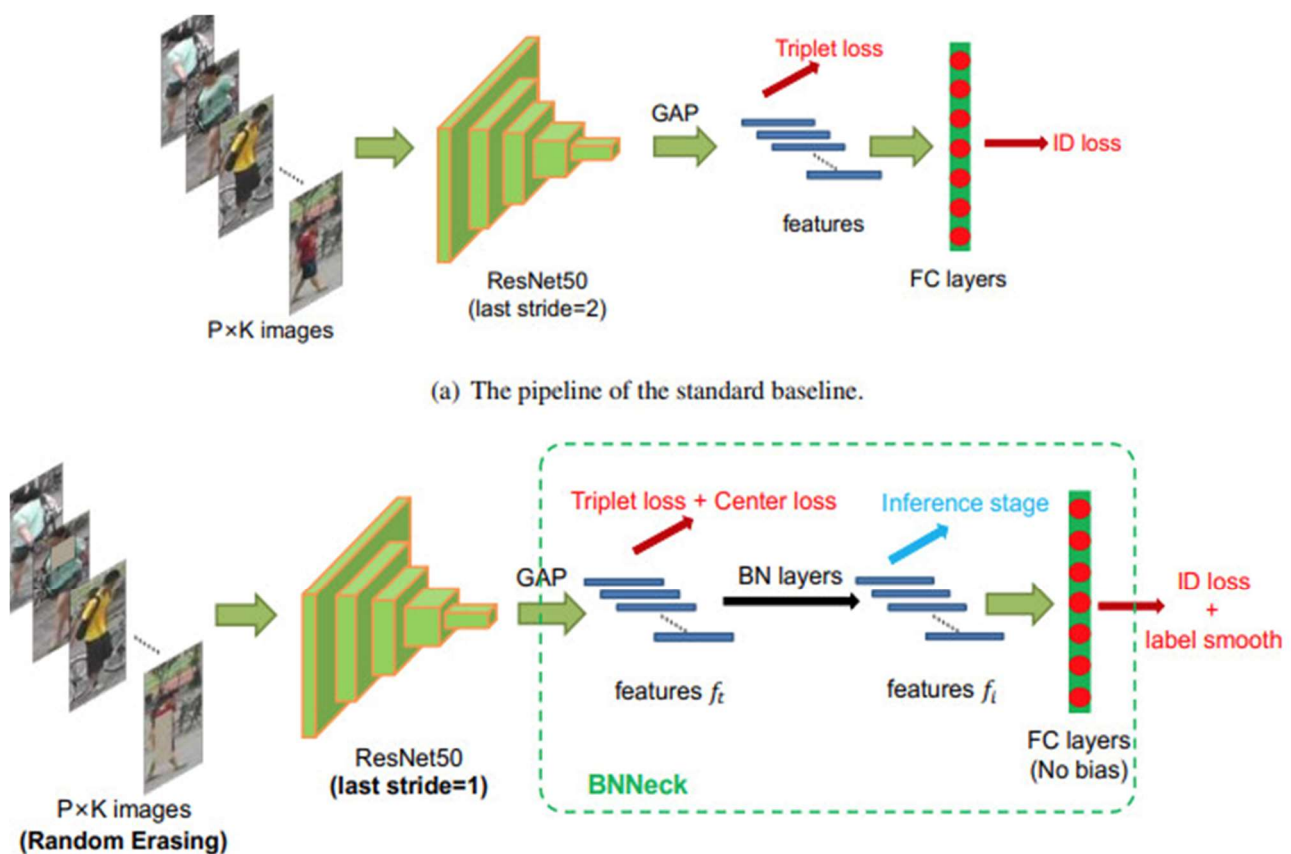
Z wykresów wynika, że potencjał do wytrenowania szybszego modelu mają wersje efficientnet-u w wariantach od b0 do b4. Ilość parametrów nie przekłada się wprost na długość obliczeń, ale jest

jednym z czynników decydującym o nim. Potraktowano to jako wstępną selekcję wersji sieci wybranych do przetestowania.

Kwestią decydującą o wyborze sieci będzie proporcja między procentowym wzrostem prędkości przetwarzania oraz spadku jakości tworzonego embeddingu.

## 5. Rozwiązania optymalizujące proces trenowania

Zmiany poprawiające proces uczenia zaimplementowane w [\[1\] reid-strong-baseline -github](#) zostały zaprezentowane na jednej grafice umieszczonej poniżej. Wszystkie te zmiany zostaną opisane w podpunktach w tym rozdziale.



Rys 12 Ilustracja przedstawiająca proces przetwarzania danych w użytym framework-u [6]

### 5.1. Rozgrzewanie kroku uczenia (Warmup Learning Rate)

Learning Rate ma bezpośredni wpływ na to jak szybko model zmienia swoje wagi. W celu wygenerowania statystyk ułatwiających manipulowaniem parametrem Learning Rate przez

optymalizator, statuje się mechanizm nazywany "Rozgrzewaniem kroku uczenia". Polega on na poświęceniu kilku pierwszych epok na stworzeniu statystyk przy ustawionym domyślnie kroku uczenia na niskim poziomie. W praktyce zaimplementowano to w sposób:

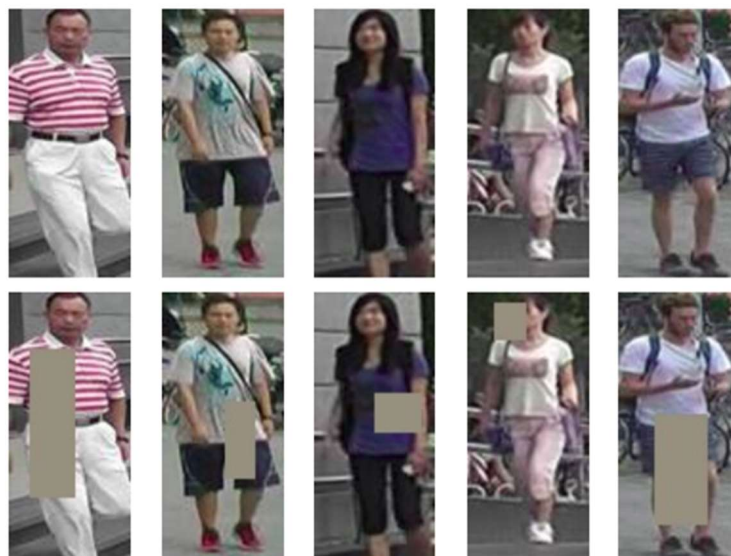
$$lr(t) = \begin{cases} 3.5 \times 10^{-5} \times \frac{t}{10} & \text{if } t \leq 10 \\ 3.5 \times 10^{-4} & \text{if } 10 < t \leq 40 \\ 3.5 \times 10^{-5} & \text{if } 40 < t \leq 70 \\ 3.5 \times 10^{-6} & \text{if } 70 < t \leq 120 \end{cases}$$

Rys 13 Ilustracja z wartościami learning rate w zależności od ilości przetrenowanych epok [6]

## 5.2. Losowo wycinanie (Random Erasing Augmentation)

Losowe wycinanie fragmentów obrazów jest powszechnie stosowanym sposobem augmentacji danych. Ma on na celu uniezależnienie modelu na zmiany, które nie powinny być ważnymi cechami obiektu na którego podstawie generowany jest jego embedding.

Przykład realizowany na opisanym powyżej datasecie Martek1501:



Rys 14 Przykłady augmentacji danych. Wycinanie losowego fragmentu obrazu [6]

## 5.3. Wygładzanie adnotacji (Label Smoothing)

Ponieważ adnotacja danych bierze udział w przeliczaniu funkcji strat jako jeden z jej składników, problemem staje się przetrenowanie modelu lub zjawisko „overconfidence”. Polega ona

na generowaniu przez model wysokiego prawdopodobieństwa podczas klasyfikacji, pomimo niskiej poprawności tej klasyfikacji.

Metodą na prewencję takiej sytuacji jest użycie "Wygładzania adnotacji". Polega to rozłożeniu niewielkiej części prawdopodobieństwa na inne klasy. W przypadku rozpatrywanym w tej pracy na bazie zbioru danych Markets1501 inną klasą jest id innej osoby.

$$q_i = \begin{cases} 1 - \frac{N-1}{N}\varepsilon & \text{if } i = y \\ \varepsilon/N & \text{otherwise,} \end{cases}$$

Id_name	Ground_truth
Alice	0
Bob	0
Frank	1
Mari	0
Eve	0

///// Label Smoothing /////

Id_name	Ground_truth
Alice	0.01
Bob	0.01
Frank	0.96
Mari	0.01
Eve	0.01

Rys 15 Przykład wygładzania adnotacji

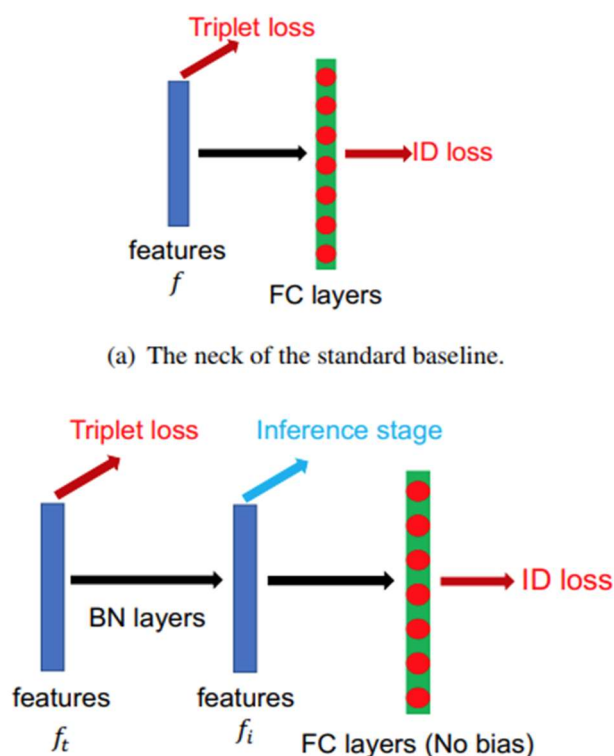
#### 5.4. Zmiana ostatniego warstwy konwolucyjnej (Last Stride)

Jedną z potrzeb jakie adresują warstwy konwolucyjne to agregacja wymiarów cech rozwiniętych przez poprzednie warstwy. Modyfikacja dotycząca tego punktu polega na ograniczeniu agregacji cech ostatniej warstwy i pozostawienia większej reprezentacji cech głębokich sieci stanowiącej backbone trenowanego modelu. Jest to zatem usunięcie ostatniej warstwy zmniejszającej rozmiar wyodrębnionych cech głębokich. Zgodnie z [6] w kontekście sieci ResNet50, w przypadku wejścia w postaci obrazu o wymiarach [256x128] ostatnia warstwa konwolucyjna zawiera stride = 2 co zmniejsza ostateczny wymiar wyniku z 16x4 do 8x2. Zmiana ta usuwa ostatnią warstwę konwolucyjną pozostawiając wynik w wymiarze 16x4.



## 5.5. BBNeck

Dużą część pracy jaką poświęcili twórcy framework-u użytego w tej pracy, stanowiła zmiana opisana jako BBNeck.



Rys 16 : Porównanie ostatnich warstw sieci w standardowym rozwiązaniu oraz BNNeck [6]

Jak zaprezentowano na załączonej ilustracji zaczerpniętej z pracy [6] w przypadku a) będącym typowym podejściem ID loss i triplet loss działają na tym samej warstwie cech głębokich (embedding-ów). W takiej sytuacji lepiej sprawdza się przeliczenie dystansu cosinusowego czyli porównanie kierunków wyznaczonych przez wektory w przestrzeni embeddingów. Natomiast w przypadku b) wykorzystanie odległości euklidesowej w przestrzeni wektorów embeddingów sprzyja zagęszczaniu wyników wewnątrz klasową oraz zwiększanie odległości między zbiorami należącymi do innych klas. Ponieważ zagęszczanie wyników wewnątrz jednej klasy może zdecydowanie szybciej minimalizować funkcję strat triple loss, niż zwiększanie odległości między klasowej wprowadzono kolejną warstwę cech głębokich. Kolejna warstwa jest znormalizowaną warstwą cech głębokich (embeddingów). Idea stojąca za takim krokiem polega na wymuszeniu stałej długości wektorów cech dla różnych klas. Zwiększa to znaczenie odległości między różnymi klasami, co prowadzi do



ograniczenia problemu minimalizacji międzyklasowej "kosztem" maksymalizacji odległości międzyklasowej. Ideę bardzo dobrze ilustruje grafika zawarta w pracy [6]

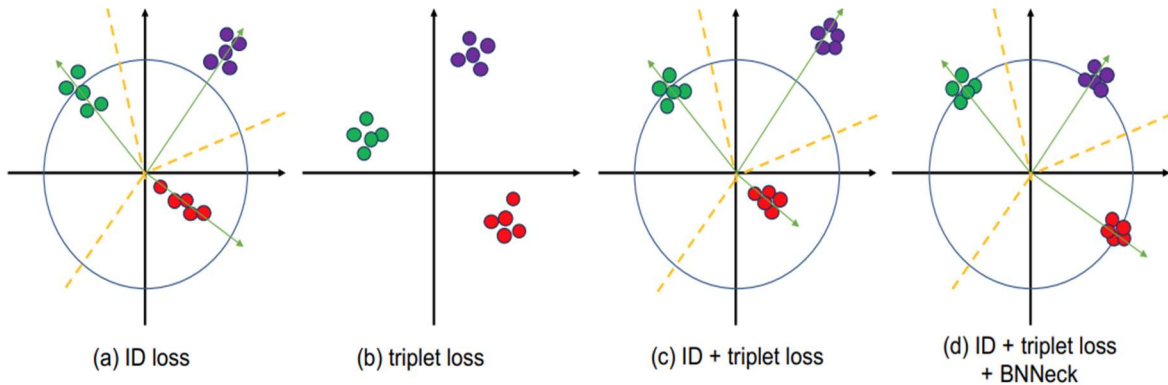


Figure 6. Two-dimensional visualization of sample distribution in the embedding space supervised by (a) ID Loss, (b) Triplet Loss, (c) ID + triplet loss and (d) ID + triplet loss + BNNeck. Points of different colors represent embedding features from different classes. The yellow dotted lines stand for the supposed classification hyperplanes.

Rys 17 Ilustracja wpływu poszczególnych rozwiązań na rozmieszczenie obrazów w przestrzeni embeddingów [6]

## 5.6. Center Loss

To rozwiązanie dodające do funkcji strat bezpośredni człon odpowiedzialny za penalizowanie dużych rozbieżności w obrębie jednej klasy. Formuła na przeliczenie takiego członu zaprezentowano poniżej:

$$\mathcal{L}_C = \frac{1}{2} \sum_{j=1}^B \left\| \mathbf{f}_{t_j} - \mathbf{c}_{y_j} \right\|_2^2,$$

gdzie:

$B$  = rozmiar batcha

$\mathbf{f}_{t_j}$  = wynik warstwy  $\mathbf{f}_t$  dla  $j$  – tego obrazu

$\mathbf{c}_{y_j}$  = wartość centrum  $y$  – owej klasy dla  $j$  obrazu.

W rezultacie funkcja strat posiada trzy człony zaprezentowane poniżej:

$$L = L_{ID} + L_{Triplet} + \beta L_C$$

gdzie:

$L$  = funkcja strat

$L_{ID}$  = człon funkcji strat generowany z klasyfikacji

$L_{Triplet}$  = człon funkcji strat generowany z "triplet – loss"

$\beta$  = waga  $L_c$

$L_c$  = człon funkcji strat generowany przez "center – loss"

## 5.7. Podsumowanie

W pracy [6] zaprezentowano wyniki jakie dają użycie poszczególnych usprawnień opisanych powyżej. To zestawienie wygląda jak poniżej:

Model	Market1501		DukeMTMC	
	r = 1	mAP	r = 1	mAP
Baseline-S	87.7	74.0	79.7	63.7
+warmup	88.7	75.2	80.6	65.1
+REA	91.3	79.3	81.5	68.3
+LS	91.4	80.3	82.4	69.3
+stride=1	92.0	81.7	82.6	70.6
+BNNeck	94.1	85.7	86.2	75.9
+center loss	94.5	85.9	86.4	76.4

Rys 18 Tabela przedstawiająca poprawę wyników dla poszczególnych technik optymalizacji treningu użytych w [6]

Jest to unikatowa wartość jaką wprowadza praca [6]. Daje ona wiarygodne informację jaką jakościową zmianę można spodziewać się z użycia każdego z usprawnień i zaplanowanie wiarygodnego porównania zmian wprowadzanych do modeli.

## 6. Rozwiązanie bazowe

W celach odniesienia dla wyników własnych modyfikacji wytrenowano model oparty o backbone Resnet50 na zbiorze Market1501. Kod oraz wyniki dodano można prześledzić pod adresem:

[https://github.com/tomektarabasz/Praca\\_Dyplomowa\\_Tomasz\\_Tarabasz/blob/0a088118d486f5aa4b61ad1d7e0788b82b181205/notebooks/PD\\_rozw\\_bazowe.ipynb](https://github.com/tomektarabasz/Praca_Dyplomowa_Tomasz_Tarabasz/blob/0a088118d486f5aa4b61ad1d7e0788b82b181205/notebooks/PD_rozw_bazowe.ipynb)

Najważniejsze z parametrów to:

<p>METRIC_LOSS_TYPE: 'triplet'</p> <p>lub</p> <p>METRIC_LOSS_TYPE: ' triplet_center '</p>	<p>Typ formułowania funkcji strat</p>
<p>OPTIMIZER_NAME: 'Adam'</p>	<p>Wybór optymalizatora.</p> <p>Optymalizator Adam bierze swoją nazwę od Adaptacyjnej estymacji momentem (Adaptive moment estimation). Wyznacza on adaptacyjnie wartość kroku uczenia dla każdego parametru.</p>
<p>MAX_EPOCHS: 120</p>	<p>Określenie maksymalnej liczby epok</p>
<p>WARMUP_FACTOR: 0.01</p>	<p>Wielkość mnożnika parametru "learning rate"</p>
<p>WARMUP_ITERS: 10</p>	<p>Ilość epok</p>
<p>WARMUP_METHOD: 'linear'</p>	<p>Metoda wykorzystana do wyznaczania parametru "learning rate"</p>
<p>IMS_PER_BATCH: 64</p>	<p>Wielkość batcha, czyli ilość obrazów ładowanych w jednej iteracji</p>
<p>STEPS: [40, 70]</p>	<p>Punkty zmiany prędkości uczenia</p>
<p>CHECKPOINT_PERIOD: 40</p>	<p>Ilość epok po których następuje zapis modelu</p>
<p>LOG_PERIOD: 20</p>	<p>Ilość epok po której następuje zapis logu z testu modelu</p>
<p>EVAL_PERIOD: 40</p>	<p>Ilość epoko po których następuje ewaluacja modelu</p>

**Treningi Resnet50 z użyciem softmax-triplet loss:**

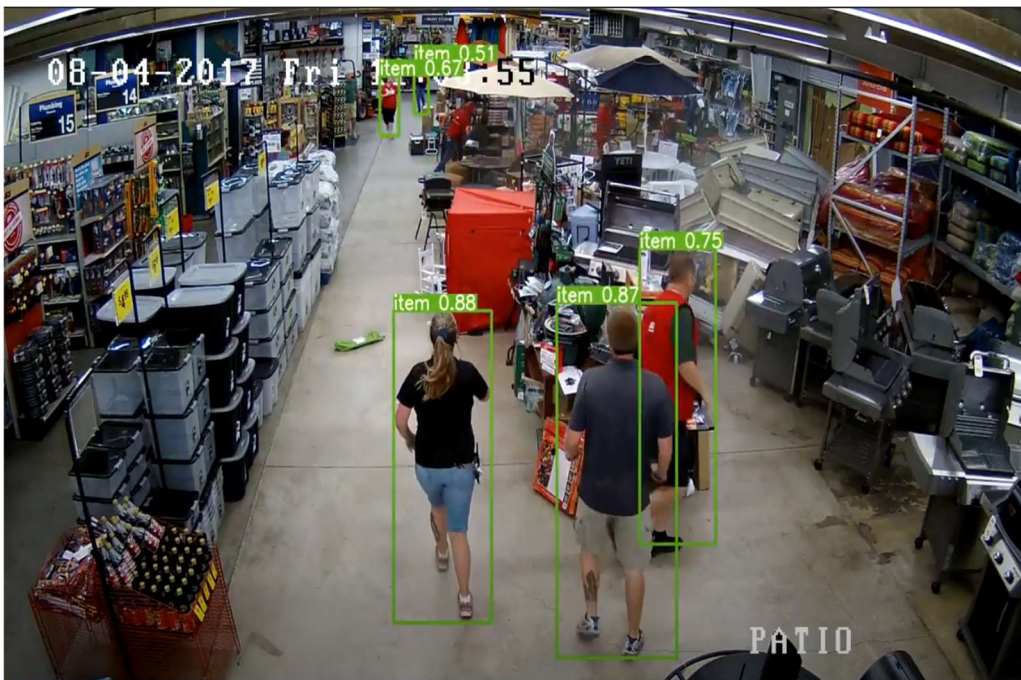
Validation Results - Epoch: 120  
mAP: 85.9%  
CMC curve, Rank-1 :94.2%  
CMC curve, Rank-5 :98.3%  
CMC curve, Rank-10 :99.0%

## 6.1. Własna ewaluacja

W celu przeprowadzenia sprawdzenia potencjału wykorzystania modelu do aplikacji śledzącej osoby na nagraniach wideo zrealizowano test na obrazach wygenerowanych z nagrania pobranego z portalu YouTube oraz własnego nagrania zrealizowanego kamerą 360. Pierwszy test miał posłużyć sprawdzeniu potencjału do odnajdywania tych samych osób na kolejnych ujęciach kamery zmieniających się w czasie. Drugi test miał natomiast posłużyć do przetestowania czasu inferencji modeli bazowych służących jako punkt odniesienia.

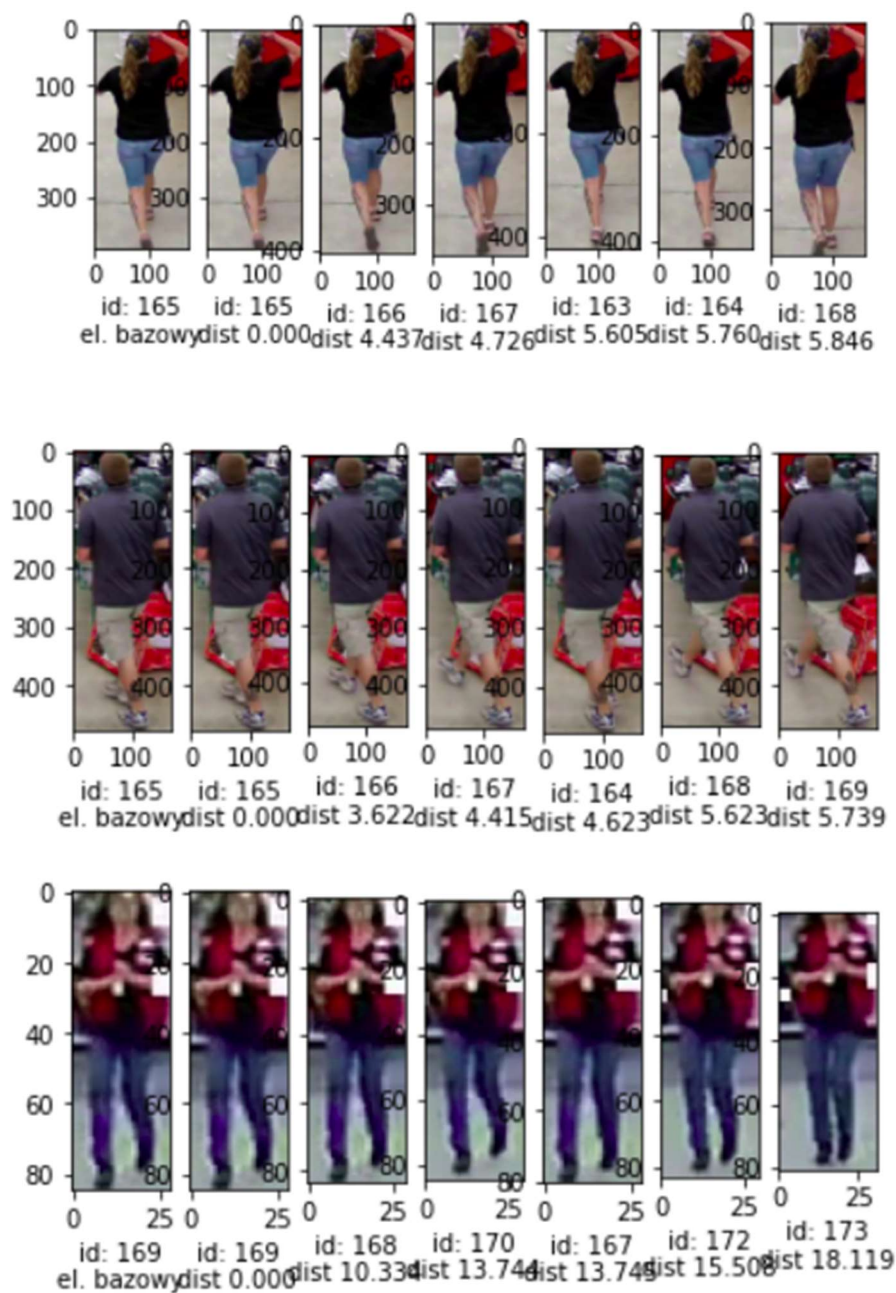
### Test 1. Próba użycia na nagraniu. Prezentacja nagrania

Zbiór wygenerowano z nagrania umieszczonego poniżej.



Korzystając z frameworku YOLO5 zmodyfikowanym na potrzeby tej pracy, jedynie do detekcji postaci ludzkich, wygenerowano dataset z wyciętymi osobami z nagrania. Posłużą one do stworzenia embeddingów i wyszukania tej samej osoby z kolejnych klatek nagrania.

**Wyniki testu zaprezentowano poniżej:**



*Rys 19 Przykładowe wykorzystanie embeddingów do śledzenia osób na nagraniu*

Są one obiecujące i dają nadzieję możliwość wykorzystania w celu śledzenia przemieszczających się osób w pomieszczeniach. Do pełnej weryfikacji potrzebne są jednak dalsze badania.

## Test 2 Sprawdzenie czasu interferencji na własnym zbiorze danych.

Wyniki zestawiono tabelarycznie:

	model_name	time	time_per_img	metric_value
<b>resnet50-no-trained</b>	resnet50-no-trained	13.955373	0.011600	0.764271
<b>resnet50_e80</b>	resnet50_e80	14.638157	0.012168	0.759869
<b>model_efficient_b0</b>	model_efficient_b0	21.347369	0.017745	0.647229
<b>model_efficient_b3</b>	model_efficient_b3	30.390721	0.025262	0.480898
<b>model_efficient_b5</b>	model_efficient_b5	42.141166	0.035030	0.478323
<b>model_efficient_b7</b>	model_efficient_b7	57.107825	0.047471	0.422554

Są one zaskakujące o tyle, że sieć EfficientNet okazuje się wolniejsza w niż ResNet50 co jest sprzeczne z dostępnymi publikacjami dotyczącymi modelu EfficientNet. Wyjaśnienie tego zagadnienia zostanie podane w dalszej części pracy.

## 7. Modyfikacje

### 7.1. Użycie jako sieci bazowej sieci EfficientNet

Główną modyfikacją jaka została wytestowana w tej pracy było użycie jako sieci bazowej do ekstrakcji cech, sieci EfficientNet. Ten model nie został zaimplementowany i przetestowany w bazowej wersji framework-u. Więcej na temat powodów wyboru tej sieci przedstawiono w punkcie 5.2.

### 7.2. Zmniejszenie rozmiaru embeddingu

W celu zmniejszenia rozmiaru generowanego embeddingu zdecydowano o zmniejszeniu rozmiaru ostatniej warstwy sieci z 2048 parametrów do 512. Ma to na celu zmniejszenie rozmiaru ostatecznie zapisywanych danych oraz przyspieszenie procesu przeszukiwania wygenerowanych wektorów.

### 7.3. Zmniejszenie precyzji zapisywanych parametrów do float16

Zmniejszenie precyzji z jaką zapisywane są parametry w sieci jest istotne dla optymalizacji procesu obliczeń na urządzeniach producenta kart graficznych Nvidia oraz dla samego rozmiaru

zapisanego modelu. Kodzie zapewniający transformację precyzji do float16 znaleźć można we własnej modyfikacji frameworku reid-strong-baseline, którą można znaleźć pod adresem:

<https://github.com/tomektarabasz/reid-strong-baseline-tt>

## 7.4. Wprowadzenie dodatkowej augmentacji danych

Dodatkowa augmentacja została już opisana w rozdziale 3. Zbiory Danych (Data sets) i wybrane metryki. Została ona wykorzystana do treningów zarówno sieci EfficientNet jak i ResNet50 poza rozwiązaniem bazowym. Nie jest to jedyna forma augmentacji, ale jest to jedyna jaką dodano do istniejącej w wykorzystywanym frameworku. Pozostałe to:

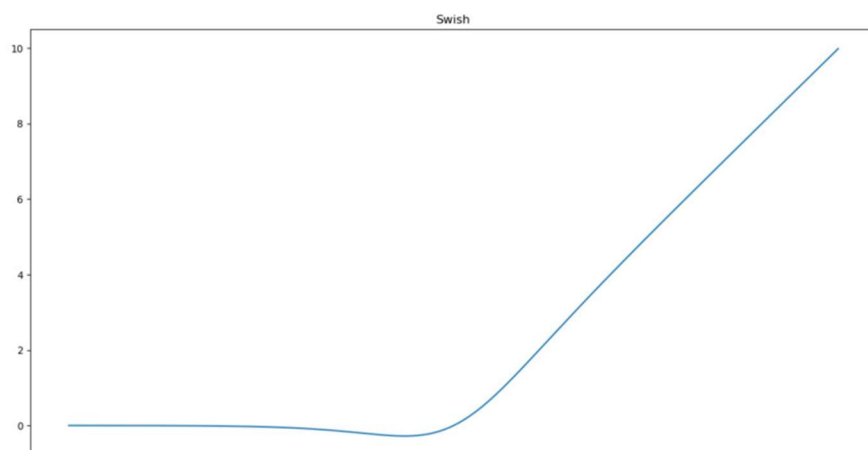
- RandomHorizontalFlip – losowe, horyzontalne odbicie
- RandomCrop – losowe przycięcie obrazu
- RandomErasing – losowe wycięcie fragmentu obrazu.

Szczegóły można sprawdzić w kodzie pod adresem:

<https://github.com/tomektarabasz/reid-strong-baseline-tt/blob/master/data/transforms/build.py>

## 7.5. Wprowadzenie funkcji Swish jako funkcji aktywacji

Funkcja Swish zdefiniowana jako:



*Rys 20 Funkcja aktywacji Swish*



Zgodnie z [\[3\] Why Swish could perform better than ReLu](#) taka postać funkcji aktywacji może poprawić wyniki poprzez:

- Dla dużych wartości ujemnych, tak jak funkcja ReLu zeruje takie aktywacje.
- Dla wartości dodatnich aktywacji funkcja Swish zachowuje się podobnie do ReLu i nie organiczna tych wartości "od góry"
- W okolicach zera funkcja jest różniczkowalna i nie zawiera nieliniowości
- Małe wartości ujemne nie są zerowane co może pomóc w doszkalaniu modelu w procesie szukania subtelnych cech głębokich.

## 8. Trening zmodyfikowanych modeli

Proces trenowania można prześledzić w notebooku:

[https://github.com/tomektarabasz/Praca\\_Dyplomowa\\_Tomasz\\_Tarabasz/blob/master/notebooks/PD\\_tren\\_zmod\\_modeli.ipynb](https://github.com/tomektarabasz/Praca_Dyplomowa_Tomasz_Tarabasz/blob/master/notebooks/PD_tren_zmod_modeli.ipynb)

oraz

[https://colab.research.google.com/drive/1PpgkPtW664Nd\\_5eOBNomEqcNjYzxVoYU?usp=sharing](https://colab.research.google.com/drive/1PpgkPtW664Nd_5eOBNomEqcNjYzxVoYU?usp=sharing)

W procesie trenowania rozpatrzono następujące przypadki:

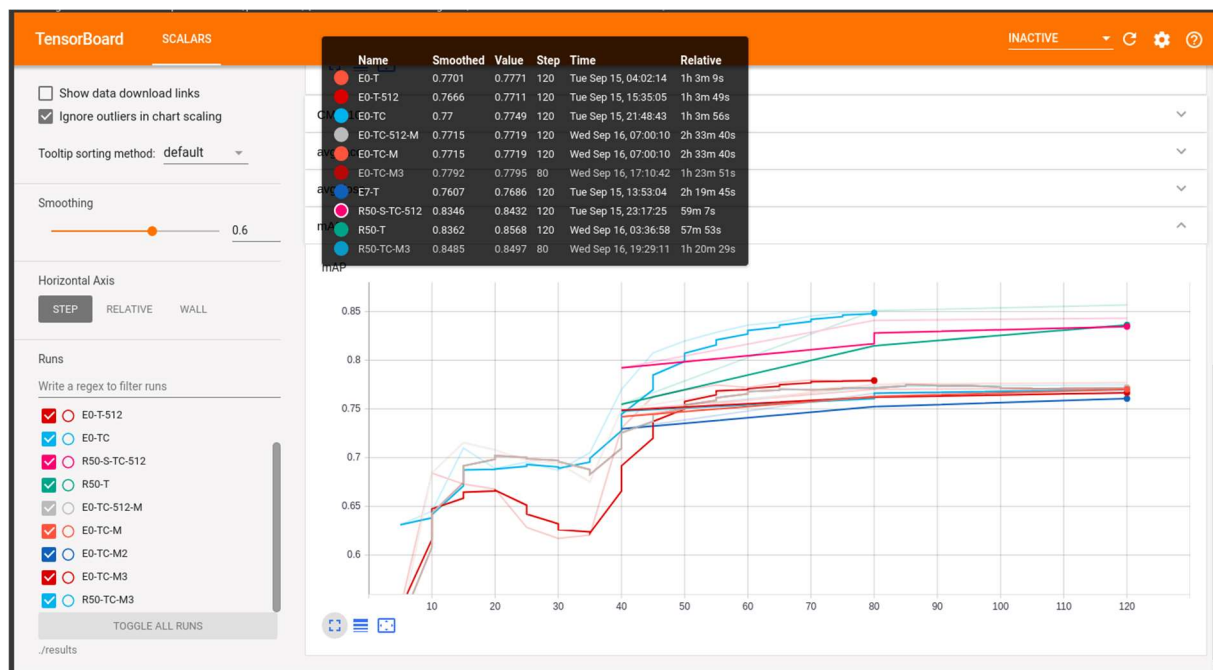
Opis testu	Rezultat
Trening EfficientNet wersja b0 z funkcją strat jako "triplet loss"	Validation Results - Epoch: 120 mAP: 77.7% CMC curve, Rank-1 :91.2% CMC curve, Rank-5 :97.4% CMC curve, Rank-10 :98.5%
Trening EfficientNet wersja b7 z funkcją strat jako "triplet loss"	Validation Results - Epoch: 120 mAP: 76.9% CMC curve, Rank-1 :90.9% CMC curve, Rank-5 :97.2% CMC curve, Rank-10 :98.3%



Trening EfficientNet wersja b0 z funkcją strat jako "triplet loss + center loss"	Validation Results - Epoch: 120 mAP: 77.5% CMC curve, Rank-1 :91.1% CMC curve, Rank-5 :97.4% CMC curve, Rank-10 :98.5%
Trening EfficientNet wersja b0 z funkcją strat jako "triplet loss + center loss" oraz zmniejszonym embeddingiem do 512 wymiarów	Validation Results - Epoch: 120 mAP: 77.2% CMC curve, Rank-1 :90.9% CMC curve, Rank-5 :97.4% CMC curve, Rank-10 :98.3%
Trening EfficientNet wersja b0 z funkcją strat jako "triplet loss + center loss" oraz zmniejszonym embeddingiem do 512 wymiarów + zwiększeniem "batch size" do 124 (2x) + zwiększenie kroku uczenia.	Validation Results - Epoch: 80 mAP: 77.9% CMC curve, Rank-1 :91.4% CMC curve, Rank-5 :97.4% CMC curve, Rank-10 :98.6%
Trening ResNet50 z funkcją strat jako "triplet loss + center loss" oraz zmniejszonym embeddingiem do 512 wymiarów + zwiększeniem "batch size" do 124 (2x) + zwiększenie kroku uczenia.	Validation Results - Epoch: 80 mAP: 85.0% CMC curve, Rank-1 :93.5% CMC curve, Rank-5 :98.2% CMC curve, Rank-10 :98.8%
Trening ResNet50 z funkcją strat jako "triplet loss + center loss" ze zmianą funkcji aktywacji z ReLu na Swish	Validation Results - Epoch: 120 mAP: 84.3% CMC curve, Rank-1 :93.8% CMC curve, Rank-5 :98.1% CMC curve, Rank-10 :98.9%

## 9. Wnioski

### 9.1. Wnioski na podstawie mAP



Legenda:

- E0 => EfficientNet b0
- E7 => EfficientNet b7
- R50 => ResNet50
- S => funkcja aktywacji Switch()
- T => funkcja strat "triplet loss"
- TC => funkcja strat "triplet loss + center loss"
- 512 => zmniejszenie embedding do 512
- M => modyfikacja procesu trenowania (dodanie własnej augmentacji)
- M2 => modyfikacja procesu trenowania (zwiększenie batch-a do 128)
- M3 => modyfikacja procesu trenowania M1 + M2

Wnioski płynące z przeprowadzonych badań zaprezentowano w punktach:

- Sieć EfficientNet wytrenowana na zbiorze ImageNet startuje o w przybliżeniu 10% gorszym wynikiem od sieci ResNet50.
- Proces trenowania przebiega podobnie dla obu sieci utrzymując w przybliżeniu stały 10% dystans między wynikami
- Zmniejszenie wielkości embeddingów z 2048 do 512 nie wpływa znacząco na uzyskane wyniki mAP
- Użycie funkcji aktywacji Swish nie poprawiło wyników modelu ResNet50

- Wykorzystanie dodatkowej augmentacji i zwiększenie batch size do 128 obrazów poprawia wyniki uczenia.

Wynika z nich, że testowana sieć EfficientNet daje gorsze wyniki niż sieć ResNet50, jest jednak lepsza od sieci ResNet156. Test odbywał się na treningowej bazie danych Market1501. Sprawdzono również na własnym zbiorze danych zbieżność generowanej reprezentacji, która daje podstawy do twierdzenia, że opisywany poprzednio test odnosi się również do poziomu generalizacji modelu. Wskazane jest jednak pogłębione sprawdzenie wytrenowanych modeli na innym pełnym zbiorze danych.

## 9.2. Wnioski na podstawie czasu interferencji

	model_name	time	time_per_img	metric_value
<b>R50_S_T_512</b>	R50_S_T_512	14.383057	0.011956	0.843891
<b>resnet50_e120</b>	resnet50_e120	15.148581	0.012592	0.816209
<b>model_efficient_b0_triplet_512</b>	model_efficient_b0_triplet_512	21.899033	0.018204	0.775219
<b>resnet50_e80</b>	resnet50_e80	14.581806	0.012121	0.759869
<b>resnet50-no-trained</b>	resnet50-no-trained	13.951077	0.011597	0.730149
<b>model_efficient_b0</b>	model_efficient_b0	21.688285	0.018028	0.647229
<b>model_efficient_b3</b>	model_efficient_b3	30.395038	0.025266	0.480898
<b>model_efficient_b5</b>	model_efficient_b5	42.966256	0.035716	0.478323
<b>model_efficient_b7</b>	model_efficient_b7	57.766054	0.048018	0.422554

Legenda:

- model\_efficient\_b(od 0 do 7)=> modele przed trenowaniem
- model\_efficient\_bo\_triplet\_512 => wykorzystanie modelu EfficientNet b0 wytrenowanego na funkcji strat "triplet loss" z embeddingiem rozmiaru 512
- R50\_S\_T\_512 => model ResNet50 z funkcją aktywacji Swish wytrenowany na funkcji strat "triplet loss" i embeddingiem rozmiaru 512

Wyniki odnoszące się do "metric\_value" zgadzają się z wynikami z poprzedniego punktu dotyczącego mAP. Dowodzi to przewagi sieci ResNet50 nad EfficientNet w stosunku do "jakości" uzyskanej reprezentacji. Jest to potwierdzenie wyników opisanych w punkcie powyżej.

Niespodziewanie jednak model Resnet50 okazał się szybszy od modelu EfficientNet b0, pomimo tego że posiada 4 razy więcej parametrów.

Ilość parametrów nie jest jedynym czynnikiem wpływającym na czas inferencji (przetwarzania), ale jest z pewnością z nią skorelowana. Na czas przetwarzania wpływają ilości obliczeń warstw nieposiadających parametrów trenowalnych oraz sposób implementacji wykonywanych konwolucji.

Odpowiedź na te nieintuicyjne wyniki wyjaśnione zostały poprzez odpowiedź udzieloną na forum serwisu github [\[13\]](#) odpowiedź rozwinięto w dyskusji [\[14\]](#). Okazuje się, że implementacja warstw konwolucyjnych wykorzystujących parametr "group", w bibliotece PyTorch jest zdecydowanie wolniejszy od implementacji w środowisku TensorFlow. Problem ten został już podjęty przez osoby rozwijające bibliotekę PyTorch.

## **10. Podsumowanie**

### **10.1. Komentarz uzyskanych wyników**

Uzyskane wyniki potwierdzają potencjał wykorzystania modeli służących reidentyfikacji do śledzenia obiektów na nagraniach z kamer przemysłowych. Zaplanowane testy sieci EfficientNet pokazały, że sieć ta nie daje lepszych wyników niż użyta w pracy [6] sieć ResNet50. Nie udało się również potwierdzić jej szybszej inferencji w stosunku do sieci ResNet co wynikało z wewnętrznych implementacji biblioteki Pytorch dotyczących warstw konwolucyjnych. Pokazano jednak, że istnieje potencjał do poprawy wyników poprzez zmianę parametrów trenowania oraz zwiększenie ilości zastosowanych technik augmentacji danych. Wypróbowano również zmiany funkcji aktywacji z ReLu na Swish, co nie przyniosło jednak znaczącej poprawy uzyskanych wyników.

### **10.2. Dalsze badania**

Wykorzystanie modeli tworzących uproszczoną reprezentację obrazów (embeddings) ma potencjał do wykorzystania w zadaniach śledzenia postaci na nagraniach. Aby stwierdzić czy mogą wypełniać to zadanie samodzielnie należałoby stworzyć zbiór danych oparty o właściwie opisane obrazy z poszczególnych ujęć z jednego nagrania. Posłużyłoby to jako podstawa do określenia wiarygodnej miary tego jak skutecznie odnajdowane i reidentyfikowane są osoby w zadaniu śledzenia i odpowiadało na postawione pytanie. Wydaje się jednak, że niewątpliwie może to być mechanizm wspierających inne algorytmy śledzenia obiektów. Spodziewanym polem, gdzie wygenerowany embedding mógłby znacząco pomóc jest rozwiązanie sytuacji mijających się obiektów.

Kolejnymi krokami rozwijającymi tę pracę byłoby również przeprowadzenie testów wydajnościowych na środowisku niezależnym od implementacji poszczególnych warstw w wykorzystywanych bibliotekach. Poprzedzone pełnym procesem optymalizacji modelu, technikami takimi jak "pruning" i "quantization" wyznaczenie wartości FPS pozwalającej analizować nagranie w czasie rzeczywistym.

Potencjał do dalszej pracy daje możliwość dodania dodatkowej augmentacji danych, które zamiast modyfikować przypadkowym szumem, modyfikowałyby obrazy wytrenowaną uprzednio maską w myśl techniki ataku na modele nazywanym "adversarial attack".

## 11. Bibliografia

Interaktywna forma bibliografii znajduje się pod adresem:

[https://github.com/tomektarabasz/Praca\\_Dyplomowa\\_Tomasz\\_Tarabasz/blob/master/notebooks/PD\\_bibliografia.ipynb](https://github.com/tomektarabasz/Praca_Dyplomowa_Tomasz_Tarabasz/blob/master/notebooks/PD_bibliografia.ipynb)

[1] reid-strong-baseline -github

[2] Mingxing Tan, Quoc V. Le: “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, 2019, International Conference on Machine Learning, 2019; arXiv:1905.11946

[3] Why Swish could perform better than ReLu

[4] Torchreid: Library for Deep Learning Person Re\_Identityfication in Pytoch

[5] Torchreid: Library for Deep Learning Person Re\_Identityfication in Pytoch - github project

[6] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, Wei Jiang: “Bag of Tricks and A Strong Baseline for Deep Person Re-identification”, 2019

[7] Market1501 - pdf

[8] GPipe - pdf

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun: “Deep Residual Learning for Image Recognition”, 2015

[10] Duke MTMC dataset

[11] YOLO5

[12] EfficientNet github

[13] EfficientNet-b0 slower than Resnet-50?

[14] The inference speed is much slower than original TensorFlow code

## 12. Spis rysunków

Rys 1 Zestawienie ilości publikacji dotyczących reidentyfikacji w latach.[6] .....	4
Rys 2 : Zestawienie wyników uzyskanych w pracy [6].....	7
Rys 3 Cykl tworzenia systemów data mining CRISP DM <a href="https://media2.picsearch.com/is?0y5BVmUtw6Ychs3zIkqkKxe79hNRgDbshMID-namuPY&amp;height=309x">https://media2.picsearch.com/is?0y5BVmUtw6Ychs3zIkqkKxe79hNRgDbshMID-namuPY&amp;height=309x</a> .....	8
Rys 4 Przykład danych ze zbioru Markets1501 .....	10
Rys 5 Przykład ujęcia z autorskiego nagrania kamerą 360 .....	11
Rys 6 Przykłady wyfiltrowanych obiektów z nagrania kamerą 360 .....	12
Rys 7 Przykład ujęcia z nagrania pobranego z portalu YouTube .....	12
Rys 8 Krzywa do wyznaczani mAP <a href="https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173">https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173</a> .....	17
Rys 9 Przykład bloku rezydualnego <a href="http://d2l.ai/chapter_convolutional-modern/resnet.html">http://d2l.ai/chapter_convolutional-modern/resnet.html</a> .....	18
Rys 10 Zestawienie różnych sieci neuronowych na podstawie ilości parametrów oraz wyniku na zbiorze danych ImageNet.....	19
Rys 11 Zestawienie modeli pod względem ilości parametrów .....	20
Rys 12 Ilustracja przedstawiająca proces przetwarzania danych w użytym framework-u [6] .....	21
Rys 13 Ilustracja z wartościami learning rate w zależności od ilości przetrenowanych epok [6].....	22
Rys 14 Przykłady augmentacji danych. Wycinanie losowego fragmentu obrazu [6].....	22
Rys 15 Przykład wygładzania adnotacji .....	23
Rys 16 : Porównanie ostatnich warstw sieci w standardowym rozwiązaniu oraz BNNeck [6].....	24
Rys 17 Ilustracja wpływu poszczególnych rozwiązań na rozmieszczenie obrazów w przestrzeni embeddingów [6] .....	25
Rys 18 Tabela przedstawiająca poprawę wyników dla poszczególnych technik optymalizacji treningu użytych w [6] .....	26
Rys 19 Przykładowe wykorzystanie embeddingów do śledzenia osób na nagraniu .....	29
Rys 20 Funkcja aktywacji Swish .....	31

