uzupełnić słowa kluczowe

UNIWERSYTET GDAŃSKI
Wydział Matematyki, Fizyki i Informatyki

Dariusz Adrychowski
nr albumu: 251682

Wizualna inwentaryzacja znaków drogowych

Uzupełnić tytuł po uzgodnieniu

Praca magisterska na kierunku:
INFORMATYKA
Promotor:
dr Jakub Neumann

Gdańsk 2018/2019

Streszczenie

Dodać streszczenie

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Słowa kluczowe

słowa kluczowe

Spis treści

Wprowadzenie 11

1. Bazowe definicje 13
1.1. Sieci neuronowe i algorytmy 13
1.1.1. R—CNN Region—Convolutional Neuron Network . . . 13
1.1.2. SSD — Single Shot MultiBox Detector 14
1.1.3. Framework'i 14
1.1.4. Architektura sieci 15
1.1.5. Wykrywanie vs klasyfikacja 17
1.2. Transfer Learning 18

2. Istniejące rozwiązania i historia rozwoju detekcji obiektów w obrazie wideo 21
2.1. Detekcja obiektów w obrazie wideo 21
2.2. Rozwiązania i narzędzia 25
2.2.1. Algorytm Viola—Jones 25
2.2.2. Dalal i Triggs - HOG 25
2.2.3. CNN 25
2.2.4. R—CNN 26
2.2.5. YOLOv3 26
2.2.6. SSD 26
2.2.7. Capsule Networks 26
2.2.8. Tensorflow 26
2.2.9. Detectron 26
2.2.10. Darknet 26
2.2.11. Mask RCNN 26
2.2.12. OpenVINO toolkit 26
2.2.13. OpenCV 26
2.2.14. Python 26

2.2.15. OpenStreet Map . . . 26
2.2.16. GIS Server . . . 26
2.3. Istniejące rozwiązania . . . 26
2.3.1. Historia . . . 26
2.3.2. NanoNets[6] . . . 27

3. Pre—procesing danych . . . 29
3.1. Pozyskanie danych . . . 29
3.2. Detekcja obiektów . . . 30
3.3. Uczenie sieci i weryfikacja rezultatów . . . 30

4. Realizacja . . . 31
4.1. Pozyskanie danych wideo . . . 31
4.2. Rozpoznawanie obiektów . . . 31
4.2.1. Porównanie skuteczności algorytmów . . . 31
4.2.2. Wybór środowiska . . . 31
4.2.3. Wybór hardware . . . 31
4.2.4. Wybór software . . . 31
4.2.5. Opis realizacji softwarowej . . . 31

5. Analiza wyników . . . 33
5.1. Dokładność działania algorytmu . . . 33
5.2. Możliwe udoskonalenia . . . 33
5.3. Wnioski końcowe . . . 33

6. Zakończenie . . . 35
6.1. Ciąg dalszy — perspektywy rozwoju . . . 35
6.2. Co dalej... . . . 35

7. Brudnopis — do USUNIECIA w pracy finalnej . . . 37
7.1. Wykrywanie obiektów OpenCV[2] . . . 37
7.2. Umiejscowienie obiektów na mapie . . . 37

A. Tytuł załącznika jeden . . . 39

B. Tytuł załącznika dwa 41

Bibliografia 43

Spis tabel 45

Spis rysunków 47

Oświadczenie 49

Todo list

uzupełnić słowa kluczowe 1
Uzupełnić tytuł po uzgodnieniu 1
Dodać streszczenie 3
uzupełnić: Theano, Keras, Tensorflow, Detectron 14
Biblioteka i narzedzia do segmentacji obrazu 26
https://www.mobileye.com/ 26
Dokonczyc rownania 29
pozyskanie danych do nauki sieci, oprogramowanie do detekcji . . . 30
Jak uczymy, jaki zbiór danych, jaka siec, opis testów i dokładności . 30

Wprowadzenie

Celem niniejszej pracy było stworzenie oprogramowania umożliwiającego automatyczną akwizycję obrazu, jego segmentację i wykrywanie zadanych wcześniej, zdefiniowanych obiektów — znaków drogowych. Następnie korzystając z informacji o ich geolokalizacji, określenie ich położenia i jeśli okaże się to możliwe umieszczenia na mapie. Całość oprogramowania z założenia będzie realizowana przy użyciu kamery i wytrenowanej sieci neuronowej, która wykona detekcję i lokalizację na podstawie danych z kamery. Z założenia rozwiązanie nie musi działać online, dane geolokalizacji umieszczone będą wewnątrz zapisu wideo. Przykładowym zastosowaniem niniejszego oprogramowania może być automatyczna inwentaryzacja znaków drogowych.

W rozdziale 1 omówione zostały wybrane, podstawowe definicje i pojęcia wykorzystywane w niniejszej pracy. W rozdziale 2 przedstawiono krótki rys historyczny klasyfikacji i wykrywania obiektów w obrazie wideo i omówiono podstawowe metody realizacji powyższych. Rozdział 3 zawiera omówienie procesu przetwarzania danych od momentu ich pozyskania, czyszczenie po ich przygotowanie do procesu trenowania sieci. Rozdział 4 zawiera szczegółowy opis procesu tworzenia sieci — od wyboru sprzętu za pomocą którego zrealizowano zadanie, poprzez trenowanie sieci po finalne przetwarzanie i wykrywanie obiektów. Rozdział 5 zawiera opis analizy pozyskanych wyników, zawiera również krótkie rozważania na temat możliwych przyszłych prac nad udoskonaleniem pracy. Rozdział 6 zawiera podsumowanie i wnioski końcowe do jakich doszedł autor po wykonaniu niniejszej pracy.

ROZDZIAŁ 1

Bazowe definicje

Poniższy rozdział zawiera opis podstawowych pojęć używanych w niniejszej pracy — ich definicję, skróconą historię i krótki opis.

1.1. Sieci neuronowe i algorytmy

W chwili pisania tej pracy do klasyfikacji i detekcji obiektów na obrazie wykorzystywano kilka różnych sieci neuronowych. Sieci te wykorzystują kilka bazowych algorytmów — wybór regionów obrazu do analizy jest decydującym czynnikiem dla każdego z algorytmów.

1.1.1. R—CNN Region—Convolutional Neuron Network

Sieci typu R-CNN zostały stworzone w celu umożliwienia szybkiej detekcji obiektów, ich lokalizacji i klasyfikacji. Sygnałem wejściowym podawanym wytrenowanej sieci jest obraz. Odpowiedzią sieci R-CNN jest zestaw koordynat ramek (ang: bounding boxes) obejmujących wykryte obiekty oraz nazwa klasy do której je zakwalifikowano. Algorytm ewoluował z sieci CNN kolejno do:

- R—CNN
- Fast—R—CNN
- Faster—R—CNN

Głównymi problemami blokującymi możliwość wykorzystania sieci typu R—CNN do detekcji obiektów w czasie rzeczywistym między innymi:

- trenowanie sieci jest skomplikowane i zabiera bardzo dużo czasu
- trening odbywa się w wielu fazach (np.: trening klasyfikacji versus nauka wyboru odpowiednich regionów do obróbki)
- sieć jest bardzo wolna na etapie wnioskowania (szczególnie kiedy ma do czynienia z danymi, które nie były objęte szkoleniem)

1.1.2. SSD — Single Shot MultiBox Detector

Algorytm opracowany przez C. Szegedyat al.[9] na przełomie roku 2016/2017. Dedykowany do szybkiej detekcji obiektów w obrazie wideo. Nazwa algorytmu wzięła się od z opisu jego działania:

- Single Shot - lokalizacja obiektu i jego klasyfikacja wykonywana jest w jednym przebiegu sieci
- MultiBox - nazwa techniki opracowanej przez Szegedy'ego et al.
- Detector - sieć jest detektorem obiektów i wykonuje ich klasyfikacji

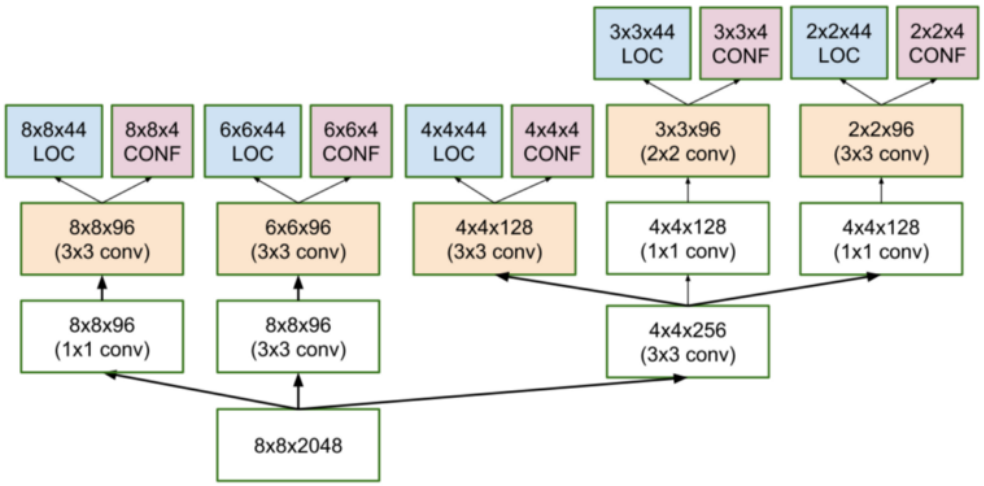
Sieci SSD budowane są w architekturze opartej na modelu VGG-16, bez warstw fully connected. Głównym powodem użycia VGG-16 jako sieci wyjściowej była wydajność tego modelu i jak również popularność (dostępność wielu modeli gotowych do transfer learning). W budowie sieci zamiast warstw fully connected zastosowano zestaw konwulcyjnych warstw pomocniczych — właściwa detekcja odbywa się w większej skali progresywnie zmniejsza rozmiar wejść do każdej następnej warstwy.

Technika MultiBox wytycza szybko koordynaty ramek (bounding boxes) otaczających obiekty bez wiedzy o ich klasie. Rys 1.1 pokazuje w jaki sposób redukowany jest liczba wymiarów ramek.

1.1.3. Framework'i

Algorytmy implementowane są w różnych wariantach i dostępne jako gotowe biblioteki w kilku frameworkách.

uzupełnić: Theano, Keras, Tensorflow, Detectron



Rysunek 1.1. Budowa wielkoskalowego konwulsyjnego przewidywania lokalizacji multiboksów. źródło: [9]

1.1.4. Architektura sieci

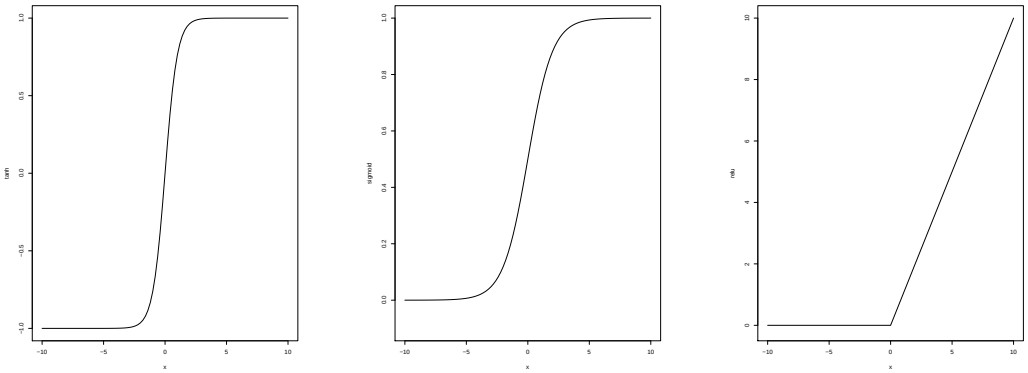
AlexNet

Historycznie biorąc AlexNet jest pierwszą z sieci neuronowych wykorzystywanych do detekcji obiektów w obrazie wideo, która była znacząco podnieść poziom dokładności wykrywania w rankingu ImageNet Classification. Składa się ona z 5 konwulcyjnych warstw i następujących po nich trzech w pełni połączonych warstw. Struktura zaproponowana przez Alexá Krizhevskyégo wyróżniała się również zastosowaniem funkcji aktywacji ReLu (Rectified Linear Unit) zamiast tradycyjnie stosowanych funkcji tangens hiperboliczny czy sigmo-idów. ReLu definiowana jest jako:

$$f(x) = max(0, x) \tag{1.1}$$

Dużą przewagą funkcji 1.1 nad poprzednio powszechnie używanymi funkcjami (tanh i sigmoid) jest szybkość uczenia się. Dzieje się tak dlatego, że po przekroczeniu pewnej wartości zarówno funkcja tanh jak i sigmoid wchodzi w stan nasycenia i zmiany impulsu wejściowego nie powodują, bądź powodują bardzo małe zmiany na wyjściu — stan ten jest nazywany problemem

zanikającego gradientu (ang.: vanishing gradient problem) Dodatkowo, aby



Rysunek 1.2. Funkcje aktywujące: kolejno tangens hiperboliczny, sigmoid, ReLu

ograniczyć przetrenowanie sieci po każdej warstwie w pełni połączonej dodano warstwę Dropout — każdy neuron z określonym prawdopodobieństwem p może zostać wyłączony.

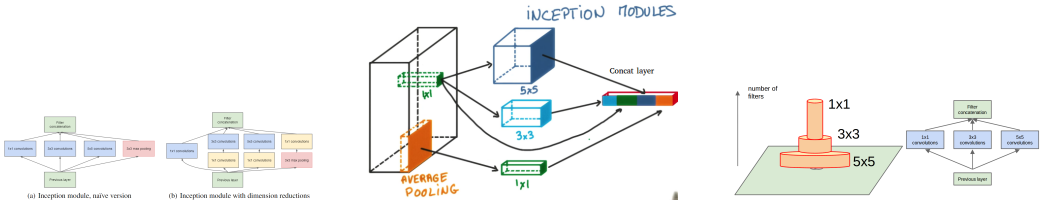
VGG—16

Grupa VGG z Oxfordu udoskonaliła AlexNet poprzez usunięcie dużych filtrów i zastąpiła je wieloma filtrami mniejszymi. Z jednej strony zmiejszyło to potrzebny nakład mocy obliczeniowej, z drugiej zwiększył głębokość sieci — co za tym idzie zdolność do nauki bardziej skomplikowanych schematów.

GoogLeNet/Inception

[12] Ponieważ VGG—16 cechowała niezwykła dokładność przy bardzo dużym nakładzie mocy obliczeniowej i pamięci potrzebnej do zaimplementowania sieci, Google rozpoczęło prace nad udoskonaleniem modelu VGG. Rezultatem tego jest właśnie GoogLeNet — cechą charakterystyczną jest warstwa wejściowa (Inception)[11]. Warstwa Inception aproksymuje rzadką sieć CNN z sieci o normalnej gęstości. Z właściwości złożonych sieci neuronowych wynika, że tylko niewielka ilość neuronów jest skuteczna – ilość filtrów jest

również redukowana. Do wychwycenia detali aplikowane są filtry o różnych wielkościach i o różnej skali.



Rysunek 1.3. GoogLeNet

Residual Networks

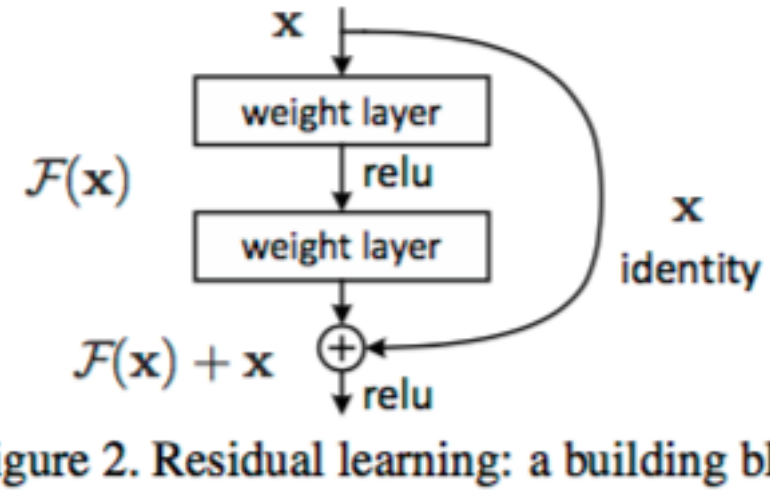
Zwiększanie głębokości sieci neuronowej (ilości warstw ukrytych) prowadzi do zwiększenia dokładności sieci jeśli zwracamy dużą uwagę na przetrenowanie sieci i do niej nie doprowadzimy. Im większa głębokość sieci, tym sygnał potrzebny do zmiany wag jest mniejszy. Maleje on wraz z ilością warstw. Łatwo zauważyć, że pierwsze warstwy tracą na ważności – zjawisko to nazywa się zanikającym gradientem. Oczywiście optymalizacja głębokiej sieci jest też dużo bardziej skomplikowana ze względu na ilość współczynników wag do dostrojenia.Problemy te częściowo rozwiązują sieci rezydualne. Sieci rezydualne usprawniają szkolenie głębokich sieci poprzez zastosowanie modułów zwanych modułami rezydualnymi.

next sub

1.1.5. Wykrywanie vs klasyfikacja

Wykrywanie

Wykrywanie obiektów wykorzystuje algorytmy klasyfikujące do określenia co i gdzie znajduje się na badanych obrazie (również wideo). Powstanie i rozwój



Rysunek 1.4. Zasada nauki sieci głębokich przy użyciu modułów rezydualnych

sieci CNN umożliwiło wyrywanie wielu klas przy pojedynczej analizie. Algorytm wykrywający (ang. object detection) po zbadaniu obrazu przedstawi rozmieszczenie na obrazie rozpoznanych obiektów.

Klasyfikacja

Klasyfikacja polega na określeniu czy obraz należy do pewnej, z góry określonej kategorii. Np. klasyfikator odpowiada na pytanie — czy na badanym obrazie znajduje się kot.

1.2. Transfer Learning

Transfer learning jest zbiorczą nazwą na zestaw technik służących do ponownego użycia i zmiany wyuczonego tematu do rozpoznawania nowego w celu przyspieszenia procesu trenowania sieci neuronowej. Ponieważ tworzenie i uczenie sieci neuronowej wymaga zaangażowania znacznych mocy obliczeniowych, a wykorzystanie transfer learning znacznie obniża to wymaganie jest to technika bardzo popularna i chętnie wykorzystywana w procesie uczenia

maszynowego. Aby przyspieszyć i ułatwić ten aspekt pracy z sieciami neuronowymi stworzono zestawy narzędzi ułatwiających to zadanie. Jako przykład posłużyć może dostępna na licencji Apache 2.0 biblioteka Xfer do pobrania z repozytorium Github[5].

Sieci neuronowe posiadają właściwość nauki korelacji pomiędzy sygnałem wejściowym (np.: obrazem) a jego reprezentacją (np.: opisem). Proces takiej nauki nazywany jest uczeniem nadzorowanym. Po wytrenowaniu sieć jest zdolna do przewidywania powiązania pomiędzy sygnałem wejściowym i najbardziej pasującą reprezentacją. Niestety, jeśli warunki brzegowe w czasie implementacji mogą ulec zmianie lub sieć ma zostać wykorzystana do wykrywania innego rodzaju danych. Np.: sieć wytrenowana do rozpoznawania samolotów w grze komputerowej — zaaplikowana w aplikacji identyfikującej samoloty z kamery operującej na prawdziwym lotnisku. Sieć może nie znać części samolotów, nie będzie ich więc w stanie rozpoznać, w świecie rzeczywistym na jakość sygnału wejściowego będą miały wpływ warunki atmosferyczne itp.. Aby zapewnić wyższą skuteczność rozpoznawania sieć powinna zostać wytrenowana na danych, na których ma pracować. Bardzo często nie jest to możliwe, bądź nie jest opłacalne ze względu na wymagany czas, nakłady mocy obliczeniowej... Możliwy i częsty jest scenariusz, w którym nie dysponujemy odpowiednio duża ilością przykładów do nauki sieci. W omawianym przykładnie może to być np.: przygotowanie się do rozpoznawania samolotów w warunkach ciężkiej zimy posiadając ograniczoną ilość zdjęć samolotów w czasie śnieżycy. W takich właśnie przypadkach wykorzystanie transfer learning jest najbardziej wskazane. Pomimo, że oryginalnie rozpoznawane obiekty i nowe są różne, posiadają one zawsze jakieś cechy wspólne (powinny takie posiadać). W takim przypadku sieć może kontynuować przerwaną naukę, bazując na elementach wspólnych wykrywanych obiektów.

Podsumowując słowami Andei'a Karpathy[7]: praktycznie niewiele osób trenuje całą sieć konwulsyjną (Convutional Network) od początku (z losową inicjlizacją wag sieci) ze względu na brak zestawu danych odpowiedniej wiel-

kości. Zamiast tego wykorzystywane są wytrenowane na dużych zestawach danych modele, które de-facto inicjalizują wagi sieci docelowej.

ROZDZIAŁ 2

Istniejące rozwiązania i historia rozwoju detekcji obiektów w obrazie wideo

Poniższy rozdział zawiera opis istniejących rozwiązań komercyjnych i niekomercyjnych, z którymi autor miał okazję zapoznać. Dodatkowo dołączono krótką historię rozwoju i opisano sposób wykorzystania ich w trakcie opracowywania rozwiązania będącego tematem niniejszej pracy.

2.1. Detekcja obiektów w obrazie wideo

„Deep Learning" koncentruje się na pięciu podstawowych domenach — klasyfikacja obrazów, rozpoznawanie mowy, semantyczna klasyfikacja tekstów i rozpoznawanie / detekcja obiektów w obrazie wideo. Z punktu widzenia formalnego wideo jest tylko sekwencją obrazów zmieniających się wraz z upływem czasu. Podejście takie zakwestionował i udokumentował Andej Karpathy — obecnie [styczeń 2019] zatrudniony jako Director of AI at Tesla. Jego praca "Large-scale Video Classification with Convolutional Neural Networks"[8] opisuje sposób detekcji obrazu wideo podobnie do detekcji obiektów CNN[1] dla modelowego obrazu. Praca ta jest jakby punktem przełomowym w dziedzinie przetwarzania obrazu — wcześniejsze rozwiązania bazowały na opisywaniu sklasyfikowanego obrazu zestawem słów go reprezentującym (klasyfikacja)[2] i decydowaniu przy użyciu algorytmu k-means oraz

[1] CNN — Convolutional Neural Networks
[2] bag of words

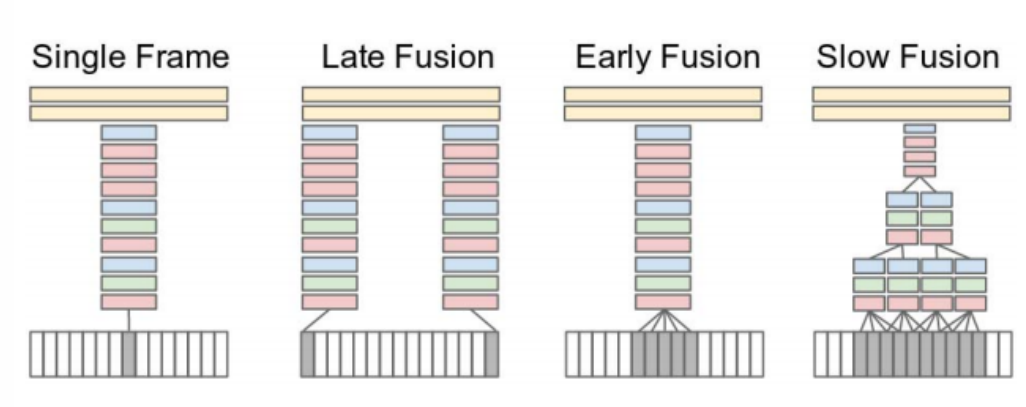
SVM [3] o treści obrazu. Artykuł przedstawia podstawy integracji wszystkich wcześniejszych technik w jeden model CNN. Podejście do analizy obrazu wideo poprzez podzielenie jej na trzy komponenty miało wpływ na wszystkie późniejsze algorytmy przetwarzania obrazu:

- połączenie elementów w dziedzinie czasu (interakcje, zmiany i przekształcenia)
- adaptatywny rozmiar analizowanego regionu
- transfer learning

Aby zmniejszyć ilość potrzebnej pamięci do budowania modelu, Karpathy ze swoim zespołem jednocześnie przetwarzał tylko jeden film, dodatkowo wstępna obróbka była przeprowadzana na różnych maszynach. Aby zapewnić jednakową długość danych wejściowych klipy pochodzące z YouTube podzielono na półsekundowe sekwencje. Agregacja przewidywanych półsekundowych klipów przypomina testowanie klasyfikacji w dziedzinie czasu - w przypadku obrazu klatka z początku klipu nawet przedstawiając ten sam obraz po pół sekundzie będzie przedstawiała ten sam obraz po pewnej transformacji. Obiekt na obu klatkach może zostać odkształcony w wyniku przemieszczenia się czy to samego obiektu czy też kamery (przekształcenia izomorficzne), jak również mogą ulec zmianie warunki w których wykonywane jest nagranie (wiatr, deszcz, nasłonecznienie itp.). Karparhy et al. przedstawiają również nowe podejście do dziedziny czasu (w momencie publikacji) — aplikują zasady opisujące sieci konwulsyjne do dziedziny czasu i zależności czasowych w wideo. Grupa ramek jest składana razem (grupowana) i staje się obrazem wejściowym do sieci CNN Standardowa sieć CNN jako wejście otrzymuje macierz danych — wysokość, szerokość i kolor (wartości w każdym z trzech składowych). Karpathy i jego zespół rozszerzyli parametry wejściowe i wcześniejszą ramkę umieścili na obecnej — zachowując rozmiary przekazują podwójną informację o kolorze — dla obu ramek osobno. Ze względu na scalanie ramek, Karpathy et al. zaproponowali różne strategie realizacji

[3] SVM — Support Vector Machines

przy założeniu klasyfikowaniu jednej ramki (kombinowanej) jednocześnie. W zależności od tego, które ramki są składane, zespół osiągnął inne rezultaty. Poniższy rysunek przedstawia przyjęte strategie.

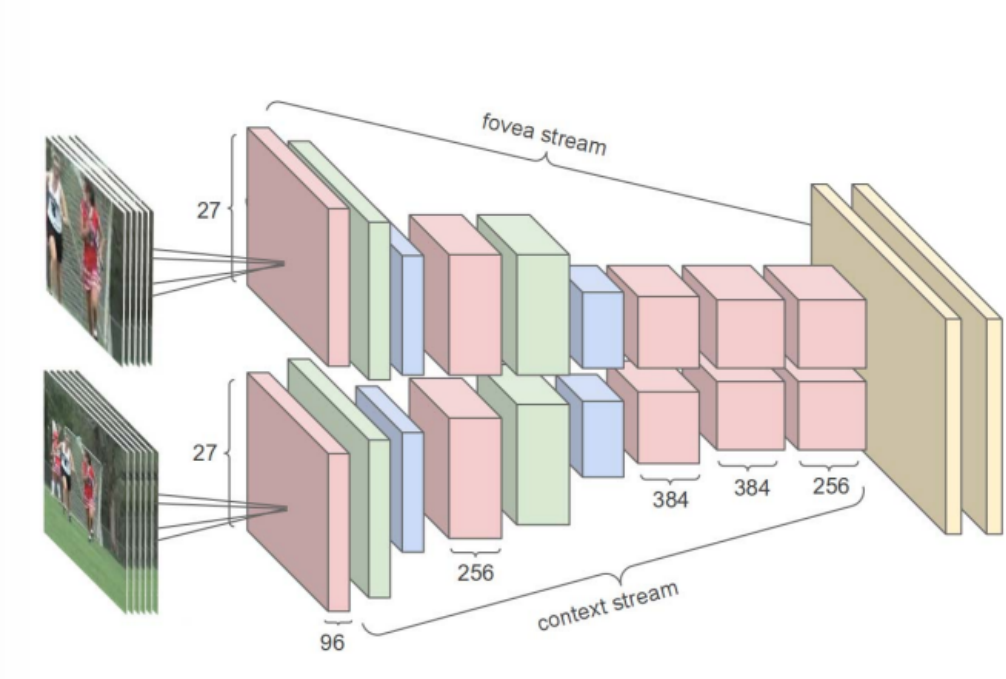


Rysunek 2.1. Strategie przetwarzania ramek zaproponowane przez Karpathy et al. źródło: "Large-scale Video Classification with Convolutional Neural Networks"

- Model Single Frame z rysunku 2.1 reprezentuje wcześniejszy sposób przetwarzania/ klasyfikacji obrazu wideo — pojedyncza ramka jest interpretowana jak statyczny obraz, informacja temporalna nie jest uwzględniana.
- Late Fusion grupuje i scala ramki skraje z zadanego przedziału czasu — na ostatnią ramkę nakłada się ramkę pierwszą.
- Early Fusion grupuje kilka ramek z ciągłego przedziału czasu.
- Slow Fusion jest najbardziej skomplikowanym modelem — cztery częściowo nakładające się na siebie grupy ramek (ovelapping równy jest dwie ramki) wybrane z ciągłego przedziału czasu są przekształcane przez sieć.

W wyniku eksperyment stwierdzono, że Slow Fusion dostarcza najbardziej użyteczną informację o obrazie, jednak jakość ta nie była dużo większa

od jakości informacji otrzymanej z modelu Single Frame. Najlepsze wyniki dawało uśrednienie informacji pochodzącej ze wszystkich rozpatrywanych modeli (Single + Early + Late + Slow). W omawianej pracy przedstawiono również koncept przetwarzana obrazu sieci CNN wielu rozdzielczości.



Rysunek 2.2. Strategia klasyfikowania obrazu przez sieci CNN o wielu rozdzielczościach źródło: "Large-scale Video Classification with Convolutional Neural Networks"

Zasada działania przedstawiona na rysunku 2.2 — dwa odrębne wejścia przetwarzają obraz przez odrębne sieci.

Pomysł przetwarzania wątków o kilku rozdzielczościach polega na jednoczesnym podaniu na wejściu odrębnych sieci neuronowych. Karpathy zaproponował dwa wątki jednocześnie – pierwszy podanie ramki o rozdzielczości 178x178 zdegradowanej do rozdzielczości 89x89 i drugiej ramki wyciętej .ze środka tej samej ramki bazowej o wymiarze 89x89. Oba wątki są sygnałem wejściowym dla odrębnych sieci o budowie Conv-MaxPool-BatchNorm.

Zauważony wzrost prędkości w porównaniu do przetwarzania jednego, nie przeskalowanego obrazu wyniósł od 2 do 4 razy.

Transfer learning.

Transfer learning polega na wykorzystaniu przetrenowanej sieci (np. jednego z dostępnych online modelów) i wykonaniu tylko końcowego dostrojenia wag sieci na nowym (docelowym) zestawie danych. Pozwala to na znaczne zaoszczędzenie czasu niezbędnego do trenowania sieci neuronowej jak również pozwala na trenowanie sieci na znacznie mniejszym zestawie danych w porównaniu do tradycyjnego trenowania sieci. Karpathy et al. wykorzystali i opisali dataset Youtube-1M do klasyfikacji zestawy danych UCF-101. Opisany eksperyment objął wykorzystanie 3 warstwowego „transfer learning". Eksperyment polegał na porównaniu wyników dokładności klasyfikacji dla różnych wariantów sieci. Porównanie umieszczono na rysunku . Jak łatwo zaobserwować transfer learning poskutkował prawie 25% wzrostem dokładności na testowanym zestawie danych. Wyniki eksperymentów przeprowadzonych przez zespół Karpathy et al został uwzględniony w niniejszej pracy.

2.2. Rozwiązania i narzędzia

2.2.1. Algorytm Viola—Jones

2.2.2. Dalal i Triggs - HOG

HOG — Histograms of Oriented Gradients (2005?)/todo[color=yellow]Sprawdzić, źródło

2.2.3. CNN

Deep Learining 2012

2.2.4. R—CNN

2.2.5. YOLOv3

YOLO - You Only Look Once [10]

2.2.6. SSD

2.2.7. Capsule Networks

2.2.8. Tensorflow

2.2.9. Detectron

2.2.10. Darknet

2.2.11. Mask RCNN

2.2.12. OpenVINO toolkit

2.2.13. OpenCV

Biblioteka i narzedzia do segmentacji obrazu

2.2.14. Python

2.2.15. OpenStreet Map

2.2.16. GIS Server

2.3. Istniejące rozwiązania

2.3.1. Historia

Historycznie rzecz biorąc pierwszym systemem rozpoznającym znaki drogowe był produkt powstały przy współpracy MobileEye

https://www.mobileye.com/

z Continental AG na potrzeby firmy BMW — system rozpoznawania znaków drogowych dla BMW serii 7. System rozpoznawania znaków od tego czasu

2.3.2. NanoNets[6]

ROZDZIAŁ 3

Pre—procesing danych

W poniższym rozdziale omówiony zostanie proces pozyskania danych, uczenia sieci neuronowej i ogólny zarys procesu powstawania.

3.1. Pozyskanie danych

Podstawowym źródłem danych dla niniejszej pracy jest kamera wideo z możliwością zakodowania pozycji geograficznej, w której znajdowała się kamera w momencie filmowania. Kamera GoPRO Hero[1] od wersji 5 poza obrazem video rejestruje dane m.in. bieżącej prędkości, lokalizacji, kierunku ruchu, przeciążenia itp. Dane te zapisywane są wraz z obrazem w strumieniu video, w formacie mp4. Korzystając z zewnętrznych narzędzi[4][3] możliwa jest ekstrakcja tych danych do jednego z otwartych formatów danych.
Obraz video kodowany jest przy użyciu kodeka w standardzie MPEG-4 Part 14 i zapisywany jest z rozszerzeniem mp4.
Po uzyskaniu dwóch ujęć tego samego obiektu, znając dokładną lokalizację miejsca skąd wykonano oba ujęcia oraz kierunek, w którym należałoby się udać z obu miejsc aby dotrzeć do rozpoznanego obiektu (a ang: bearings) można obliczyć położenie obiektu. Jak podaje źródło[13]
Formula— odległość kątowa pomiędzy p1 — p2
$\delta_{12} = 2\arcsin(\sqrt{(sin^2(\frac{\Delta\varphi}{2}) + \cos\varphi_1 * \cos\varphi_2 * sin^2(\frac{\Delta\lambda}{2}))})$
$\theta_a = \arccos(\frac{(sin\varphi_2 - sin\varphi_1 * \cos\delta_{12})}{(sin\theta_{12} * \cos\varphi_1)})$
$\theta_b = \arccos(\frac{(sin\varphi_1 - sin\varphi_2 * \cos\delta_{12})}{(sin\theta_{12} * \cos\varphi_2)})$

Dokonczyc rownania

3.2. Detekcja obiektów

pozyskanie danych do nauki sieci, oprogramowanie do detekcji

3.3. Uczenie sieci i weryfikacja rezultatów

Jak uczymy, jaki zbiór danych, jaka siec, opis testów i dokładności

ROZDZIAŁ 4

Realizacja

4.1. Pozyskanie danych wideo

4.2. Rozpoznawanie obiektów

4.2.1. Porównanie skuteczności algorytmów

4.2.2. Wybór środowiska

4.2.3. Wybór hardware

4.2.4. Wybór software

4.2.5. Opis realizacji softwarowej

ROZDZIAŁ 5

Analiza wyników

5.1. Dokładność działania algorytmu

5.2. Możliwe udoskonalenia

5.3. Wnioski końcowe

ROZDZIAŁ 6

Zakończenie

6.1. Ciąg dalszy — perspektywy rozwoju

6.2. Co dalej...

ROZDZIAŁ 7

Brudnopis — do USUNIECIA w pracy finalnej

7.1. Wykrywanie obiektów OpenCV[2]

7.2. Umiejscowienie obiektów na mapie

DODATEK A

Tytuł załącznika jeden

Treść załącznika jeden.

DODATEK B

Tytuł załącznika dwa

Treść załącznika dwa.

Bibliografia

[1] . https://community.gopro.com/t5/GoPro-Metadata-Visualization/Extracting-the-metadata-in-a-useful-format/gpm-p/40293. [Online; dostęp 22 styczeń 2019].

[2] V. Gupta. Face Detection – OpenCV, Dlib and Deep Learning. https://www.learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python, October 22$_{\text{nd}}$, 2018. [Online; dostęp 11 styczeń 2019].

[3] J. Irache. GoPro Metadata Format Parser + GPMD2CSV. https://github.com/JuanIrache/gopro-utils. [Online; dostęp 22 styczeń 2019].

[4] K. Iturbe. GoPro Metadata Format Parser. https://github.com/KonradIT/gopro-utils. [Online; dostęp 22 styczeń 2019].

[5] P. G. M. N. A. A. D. Jordan Massiah, Keerthana Elango. Xfer. https://github.com/amzn/xfer/. [Online; dostęp 6 luty 2019].

[6] G. Kaila. How to easily do Object Detection on Drone Imagery using Deep learning. https://medium.com/nanonets/how-we-flew-a-drone-to-monitor-construction-projects-in-africa-using-deep-learn 2018. [Online; dostęp 11 styczeń 2019].

[7] A. Karpathy. Cs231n convolutional neural networks for visual recognition. http://cs231n.github.io/transfer-learning/. [Online; dostęp 7 luty 2019].

[8] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[10] S. Darkflow. https://github.com/thtrieu/darkflow. [Online; dostęp 6 styczeń 2019].

[11] L. A. Santos. Artificial inteligence. https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/googlenet.html, 02 2019. [Online: 9 luty 2019].

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[13] C. Veness. Movable Type Scripts. http://www.movable-type.co.uk/scripts/latlong.html, 2019. [Online; dostęp 22 styczeń 2019].

Spis tabel

Spis rysunków

1.1. Budowa wielkoskalowego konwulsyjnego przewidywania lokalizacji multiboksów. źródło: [9] 15
1.2. Funkcje aktywujące: kolejno tangens hiperboliczny, sigmoid, ReLu 16
1.3. GoogLeNet 17
1.4. Zasada nauki sieci głębokich przy użyciu modułów rezydualnych 18

2.1. Strategie przetwarzania ramek zaproponowane przez Karpathy et al. źródło: "Large-scale Video Classification with Convolutional Neural Networks" 23
2.2. Strategia klasyfikowania obrazu przez sieci CNN o wielu rozdzielczościach źródło: "Large-scale Video Classification with Convolutional Neural Networks" 24

Oświadczenie

Ja, niżej podpisany(a) oświadczam, iż przedłożona praca dyplomowa została wykonana przeze mnie samodzielnie, nie narusza praw autorskich, interesów prawnych i materialnych innych osób.

.....	.....
data	podpis