Raport z badania tematu

„Rozpoznawanie obrazów bez użycia sieci neuronowych”

Określenie „przetwarzanie obrazu” odnosi się do szerokiej klasy problemów, dla których danymi wejściowymi są obrazy, a wyjściowymi mogą być obrazy, ale także zestawy związanych z nimi cech. Istnieje wiele wariacji tych problemów: klasyfikacja, segmentacja, adnotacja, wykrywanie obiektów i t. p. Problem klasyfikacji jest nie tylko najprostszym z nich, ale także jest podstawą wielu innych problemów z tej puli.

Ogólne podejście do problemu klasyfikacji obrazów składa się z dwóch następujących kroków:

- Generacja cech znaczących.
- Klasyfikacja obrazu na podstawie jego cech.

Do wygenerowania cech znaczących można użyć różnego rodzaju transformacji, np. konwersja obrazu na skali szarości, co zmniejsza wymiar macierzy reprezentującej obraz do 2 lub wykrywanie krawędzi, co pozwala skupić się na kształtach obiektów, histogramów, np. histogram zorientowanych gradientów lub histogramu gradientów kolorów, skalo-niezmienniczego przekształcania cech, wykrywania rogów i innych.

Histogram of Oriented Gradients (HOG)

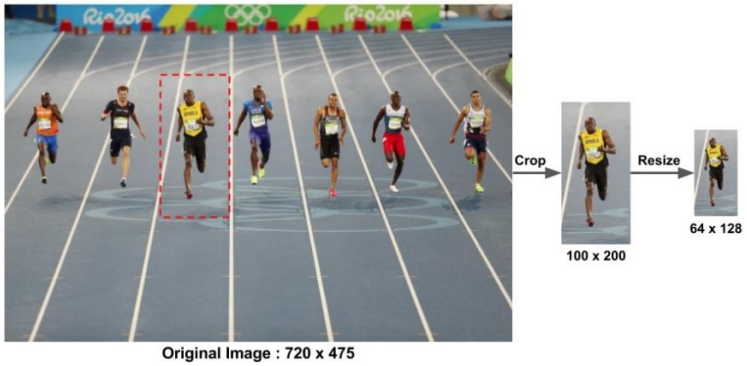
Histogram zorientowanych gradientów jest deskryptorem cech. Deskryptor cech jest to reprezentacja obrazu która upraszcza go wyróżniając użyteczną informację i wyrzucając niepotrzebną.

Zazwyczaj, deskryptor cech przekształca trójwymiarową macierz obrazu (wysokość * szerokość * RGB kanały) obraz na wektor o długości n. W przypadku HOG, zdjęcie wejściowe ma rozmiar 64 x 128 x 3, a wektor cech ma długość 3780. Można go użyć również do innych rozmiarów obrazu, jednak takie charakterystyki zostały przedstawione w oryginalnym artykule.

W deskryptorze HOG jako cechy są używane histogramy (dystrybucje) kierunków gradientów. Gradienty (pochodne po x oraz y) obrazu są użyteczne, ponieważ ich wartości są duże około krawędzi, które przechowują znacznie więcej informacji o kształcie obiektu od obszarów płaskich.

Aby obliczyć histogram zorientowanych gradientów, trzeba wykonać następujące kroki:

1. **Przetwarzanie wstępne**
   Ponieważ oryginalny algorytm był używany do wykrycia pieszych na drodze, w opisywanym przykładzie musi być zachowana proporcja 1:2. Rozmiar wycinku 64 x 128 pikseli daje wystarczająco informacji, więc będziemy używać tego rozmiaru.
   Aby znaleźć na obrazie pieszego, musimy wycinać z różnych jego miejsc kawałki o proporcji 1:2 i przekształcać ich na obrazy w rozmiarze 64 x 128 pikseli.



2. **Obliczenie gradientów**

Następnie, obliczamy gradienty po x oraz y. Dalej, obliczamy wielkość oraz kierunek gradientu używając następujących wzorów:

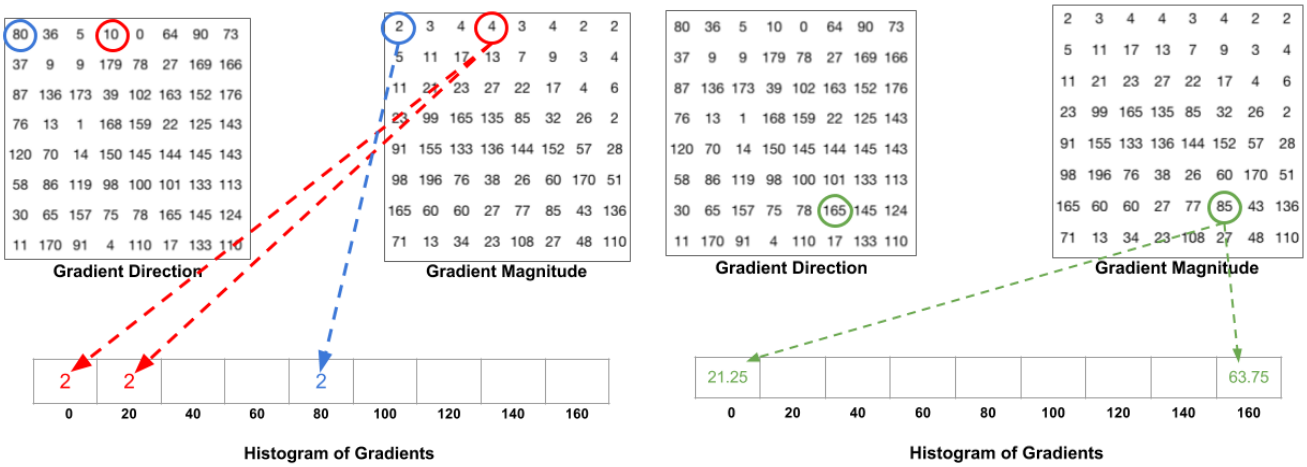
$$g = \sqrt{g_x^2 + g_y^2}$$
$$\theta = \arctan \frac{g_y}{g_x}$$

Na obrazie poniżej są przedstawione gradienty obliczone dla powyższego wycinku obrazu po x, później po y i na końcu wielkość gradientów:

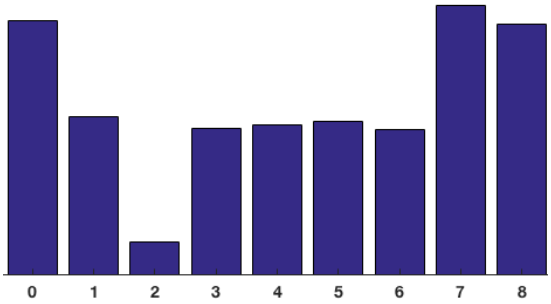
Jak widać z obrazów, gradient usunął wiele nieprzydatnych cech takich jak np. kolor tła.

3. **Obliczenie histogramu gradientów**

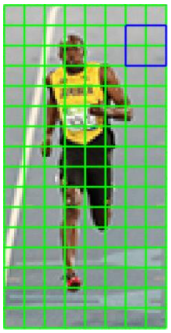
Następnie, aby zmniejszyć rozmiar oraz w celu redukcji wpływu szumów, podzielimy obraz na części o rozmiarach 8 x 8 pikseli i policzymy histogram gradientów dla każdej z tych cali.



W wyniku otrzymujemy histogram o długości 9.



4. **Normalizacja bloków**
   Dalej, aby gradient był mniej wrażliwy na całkowitą jasność obrazu, musimy podzielić obraz na kawałki o rozmiarze 16 x 16 pikseli i znormalizować każdy z 4 zawartych w nich histogramów. Otrzymamy po tym działaniu zestaw znormalizowanych wektorów o rozmiarze 36.

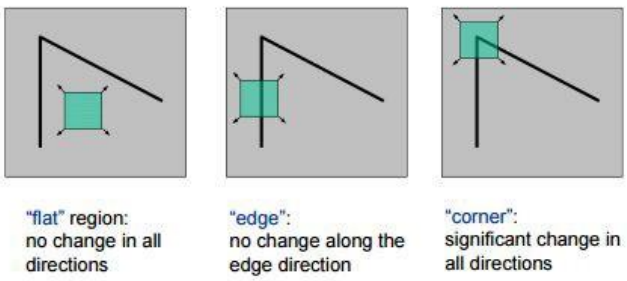


5. **Obliczenie wektora cech HOG**
   Ostatnim krokiem jest obliczenie wektora cech dla całości obrazu, w wyniku czego otrzymujemy wektor o długości 3780.

   Jeśli nałożyć obliczony przez nas wektor na obrazek, będzie to wyglądało następująco:

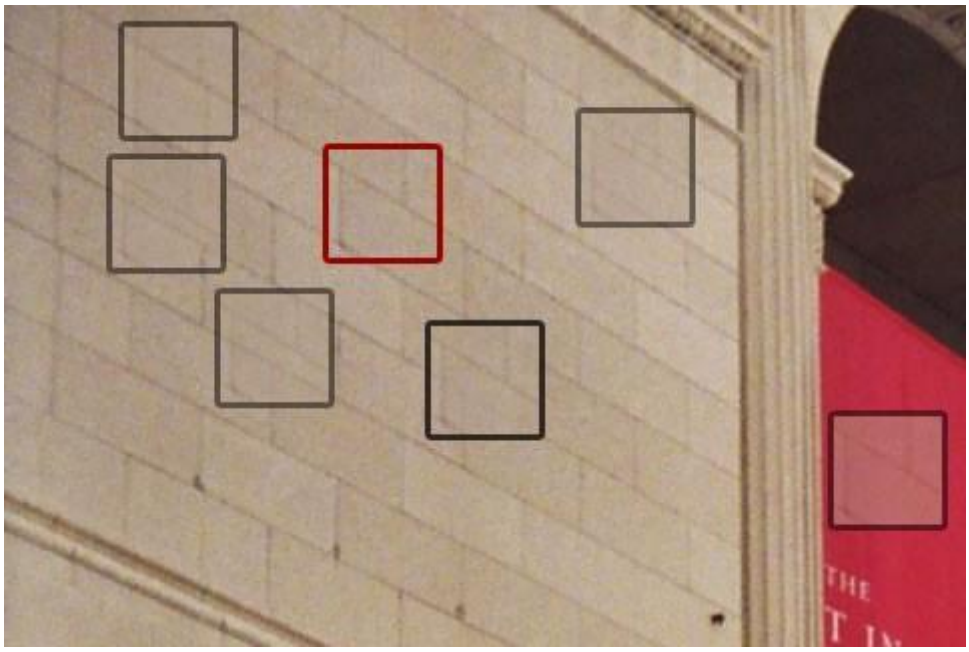
Harris Corner Detector (HCD)

Innym deskryptorem cech jest Harris Corner Detector, czyli detektor rogów. Róg to punkt, którego lokalne sąsiedztwo znajduje się w dwóch dominujących i różnych skrajnych kierunkach. Innymi słowy, narożnik można interpretować jako połączenie dwóch krawędzi, gdzie krawędź to nagła zmiana jasności obrazu. Narożniki są ważnymi elementami obrazu i ogólnie są określane jako punkty zainteresowania, które są niezmienne względem przesunięcia, obrotu i oświetlenia obrazu.



Na rysunku powyżej, jeśli weźmiemy płaski obszar, nie obserwuje się żadnej zmiany gradientu w żadnym kierunku. Podobnie w obszarze krawędziowym nie obserwuje się żadnej zmiany gradientu wzdłuż kierunku krawędzi. Tak więc zarówno region płaski, jak i region krawędziowy nie nadają się do dopasowywania łat, ponieważ nie są bardzo charakterystyczne (istnieje wiele podobnych łat wzdłuż krawędzi w obszarze krawędzi). W rejonie narożnym obserwujemy znaczną zmianę gradientu we wszystkich kierunkach. Z tego powodu rogi są uważane za dobre do dopasowywania łat (przesunięcie okna w dowolnym kierunku powoduje dużą zmianę wyglądu) i ogólnie są one bardziej stabilne przy zmianie punktu widzenia.

Chodzi o to, aby wziąć pod uwagę małe okienko wokół każdego piksela p w obrazie. Chcemy zidentyfikować wszystkie takie okna pikseli, które są unikalne. Niepowtarzalność można zmierzyć, przesuwając każde okno o niewielką wartość w danym kierunku i mierząc wielkość zmian zachodzących w wartościach pikseli.



Bierzemy sumę kwadratów różnicy wartości pikseli przed i po przesunięciu i identyfikujemy okna pikseli, w których suma kwadratów różnicy jest duża dla przesunięć we wszystkich 8 kierunkach. Zdefiniujmy funkcję zmiany ***E(u, v)*** jako sumę wszystkich sum kwadratów różnic, gdzie ***u***, ***v*** to współrzędne ***x***, ***y*** każdego piksela w naszym oknie 3 x 3, a ***I*** to wartość intensywności piksela. Cechy obrazu to wszystkie piksele, które mają duże wartości ***E(u, v)***, zgodnie z pewnym progiem.

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

Musimy zmaksymalizować tę funkcję ***E(u, v)*** do wykrywania narożników. Oznacza to, że musimy zmaksymalizować drugi człon. Stosując ekspansję Taylora do powyższego równania i używając kilku kroków matematycznych, otrzymujemy równanie końcowe:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \left(\sum \begin{bmatrix} I_x{}^2 & I_x I_y \\ I_x I_y & I_y{}^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

Zmieńmy nazwę sumarycznej macierzy na M:

$$M = \sum w(x,y) \begin{bmatrix} {I_x}^2 & I_x I_y \\ I_x I_y & {I_y}^2 \end{bmatrix}$$

Ostatecznie, równanie będzie miało następującą postać:

$$E(u,v) \approx [u \quad v]\, M \begin{bmatrix} u \\ v \end{bmatrix}$$

Ponieważ chcemy, aby suma kwadratów różnicy była duża w przesunięciach dla wszystkich ośmiu kierunków, lub odwrotnie, aby suma kwadratów różnicy była mała dla żadnego z kierunków, rozwiązując wektory własne ***M***, możemy otrzymać kierunki zarówno dla największych, jak i najmniejszych wzrostów sumy kwadratów różnicy. Odpowiednie wartości własne dają nam rzeczywistą wartość tych wzrostów. Wynik ***R*** jest obliczany dla każdego okna:

$$R = \det M - k(trace\, M)^2$$

$$\det M = \lambda_1 \lambda_2$$

$$trace\, M = \lambda_1 + \lambda_2$$

λ1 i ***λ2*** są wartościami własnymi ***M***. Zatem wartości tych wartości własnych decydują o tym, czy region jest rogiem, krawędzią czy płaskim.

- Kiedy ***|R|*** jest mała, co ma miejsce, gdy ***λ1*** i ***λ2*** są małe, region jest płaski.
- Kiedy ***R*** < 0, co ma miejsce, gdy ***λ1*** >> ***λ2*** lub odwrotnie, region jest krawędzią.
- Kiedy ***R*** jest duże, co ma miejsce, gdy ***λ1*** i ***λ2*** są duże, a ***λ1***∼***λ2***, region jest rogiem.

Generalnie algorytm HCD działa następująco:

1. Konwertuje oraz na skali szarości.
2. Stosuje filtr Gaussa, aby wygładzić szumy.
3. Stosuje operator Sobela, aby znaleźć wartości gradientu ***x*** i ***y*** dla każdego piksela obrazu w skali szarości.
4. Dla każdego piksela p na obrazie w skali szarości rozważa okno 3 × 3 wokół niego i oblicza funkcję siły rogu co nazywa się wartością Harrisa.
5. Znajduje wszystkie piksele, które przekraczają określony próg i są lokalnymi maksimami w określonym oknie (aby zapobiec redundantnemu duplikowaniu funkcji).
6. Dla każdego piksela spełniającego kryteria podane w punkcie 5 oblicza deskryptor funkcji.

Color Gradient Histogram

Histogram przedstawia rozkład kolorów na obrazie. Można to zwizualizować w postaci wykresu, który będzie pokazywał rozkład intensywności (wartości pikseli). Jeżeli np. przyjąć przestrzeń kolorów RGB, to te wartości pikseli będą mieściły się w zakresie od 0 do 255. W innej przestrzeni kolorów, zakres pikseli może być inny.

Podczas tworzenia histogramu oś X służy jako „pojemniki”. Jeśli skonstruujemy histogram z 256 przedziałami, to skutecznie policzymy, ile razy występuje każda wartość piksela. Z drugiej strony, jeśli używamy tylko 2 (równo rozmieszczonych) pojemników, to liczymy, ile razy piksel znajduje się w zakresie [0, 128) lub [128, 255]. Liczba pikseli podzielonych według wartości na osi X jest następnie wykreślana na osi Y.

Wystarczy zbadać histogram obrazu, aby uzyskać ogólne informacje dotyczące kontrastu, jasności i rozkładu intensywności tego obrazu. Histogramy te mogą więc służyć jako wektory cech (tj. lista liczb używanych do ilościowego określenia obrazu i porównania go z innymi obrazami). Zakładamy, że obrazy o podobnych rozkładach kolorów są semantycznie podobne.

Porównanie „podobieństwa” histogramów kolorów można przeprowadzić za pomocą metryki odległości. Typowe opcje to: odległość Euklidesowa, korelacja, Chi-squared, przecięcie oraz Bhattacharyja.

Algorytmy rozpoznawania obrazów

Spośród wszystkich algorytmów rozpoznawania obrazów bez użycia sieci neuronowych, wybraliśmy trzy najbardziej popularne metody rozwiązania problemu klasyfikacji. Są to algorytmy k-Nearest Neighbours, Support Vector Machine oraz Box of Words.

kNN - *k-Nearest Neighbours*

Klasyfikator k-Nearest Neighbours jest zdecydowanie najprostszym maszynowym algorytmem nauki/ klasyfikacji obrazu. W rzeczywistości jest on tak prosty, że tak naprawdę niczego się nie "uczy".

Wewnątrz, algorytm ten po prostu opiera się na odległości pomiędzy wektorami cech, podobnie jak w przypadku budowania wyszukiwarki obrazów - tylko tym razem, mamy etykiety związane z każdym obrazem, więc możemy przewidzieć i zwrócić rzeczywistą kategorię dla obrazu.

Mówiąc najprościej, algorytm k-NN klasyfikuje nieznane punkty danych, znajdując najczęstszą klasę wśród k-najbliższych przykładów. Każdy punkt danych w k najbliższych przykładach oddaje głos, a wygrywa kategoria z największą liczbą głosów!

Aby zastosować klasyfikację k-Nearest Neighbours, musimy zdefiniować funkcję metryki odległości lub podobieństwa. Do typowych wyborów należy obliczyć odległość euklidesowa:

$$d(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{\sum_{i-1}^{N} (q_i - p_i)^2}$$

Oraz Manhattan distance:

$$d(\boldsymbol{p}, \boldsymbol{q}) = \sum_{i-1}^{N} |q_i - p_i|$$

W zależności od rodzaju danych można użyć innych funkcji metryki odległości/podobieństwa (dla rozkładów [tj. histogramów] często stosuje się odległość chi-squared distance).

Na przykładzie Kaggle Dogs vs Cats dataset była zbadana efektywność i dokładność tego algorytmu przy k=1.

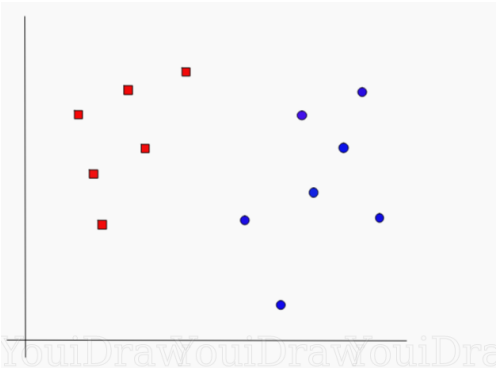
„Wykorzystując tylko surową intensywność pikseli obrazu wejściowego, uzyskaliśmy dokładność 54,42%. A dzięki zastosowaniu histogramów kolorystycznych, uzyskaliśmy nieco lepszą dokładność 57.58%. Ponieważ oba te wyniki są > 50% (powinniśmy spodziewać się 50% dokładności po prostu przypadkowo zgadując), możemy stwierdzić, że w surowych pikselach/histogramach kolorystycznych znajduje się wzorzec, który może być użyty do rozróżniania psów od kotów (chociaż dokładność 57% jest dość słaba).”

Na podstawie tych wyników można stwierdzić, że kNN bardzo zależy od hyperparametrów. Dlatego do uzyskania lepszych wyników trzeba będzie dostosowywać te hyperparametry.

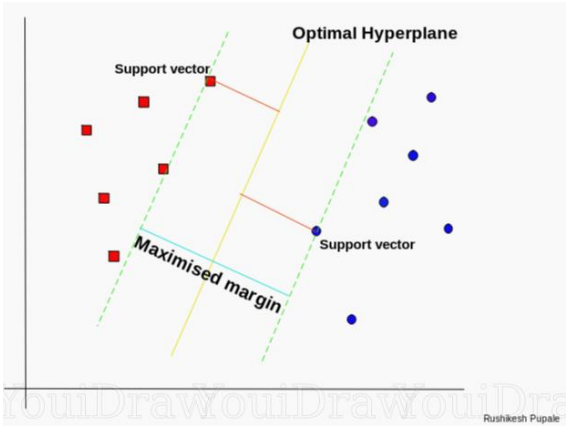
SVM – Support Vector Machine

To jest algorytm uczenia się z nauczycielem (supervised)**,** głównym jego zadaniem jest znalezienie najbardziej poprawnej linii, czyli hiperpłaszczyzny w przestrzeni, która dzieli dane na dwie klasy. SVM odbiera dane na wejściu i zwraca tę linię oddzielającą.

Np. mamy zbiór danych i chcemy sklasyfikować i podzielić czerwone kwadraty z niebieskich kółek (powiedzmy pozytywnych i negatywnych). Głównym celem w tym zadaniu będzie znalezienie "idealnej" linii, która podzieli te dwie klasy.



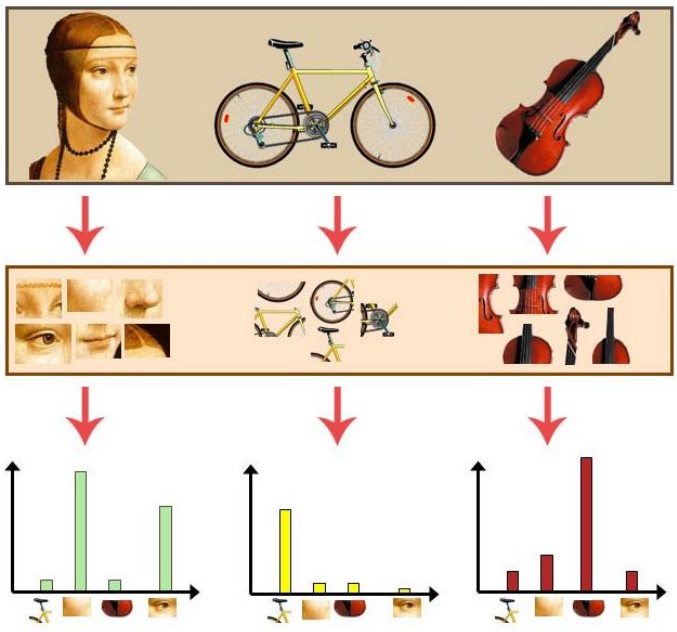
Ale problem jest w tym że nie ma jednej, niepowtarzalnej linii, która rozwiązałaby taki problem. Możemy znaleźć nieskończoną ilość takich linii, które mogą podzielić te dwie klasy.



SVM wyszukuje punkty na wykresie, które znajdują się najbliższej do linii podziału. Punkty te są nazywane wektorami odniesienia. Następnie algorytm oblicza odległość między wektorami odniesienia a płaszczyzną oddzielającą. Jest to odległość, która nazywana jest szczeliną. Głównym celem algorytmu jest zmaksymalizowanie odstępu. Za najlepszą hiperpłaszczyznę uważa się hiperpłaszczyznę, dla której szczelina jest największa.

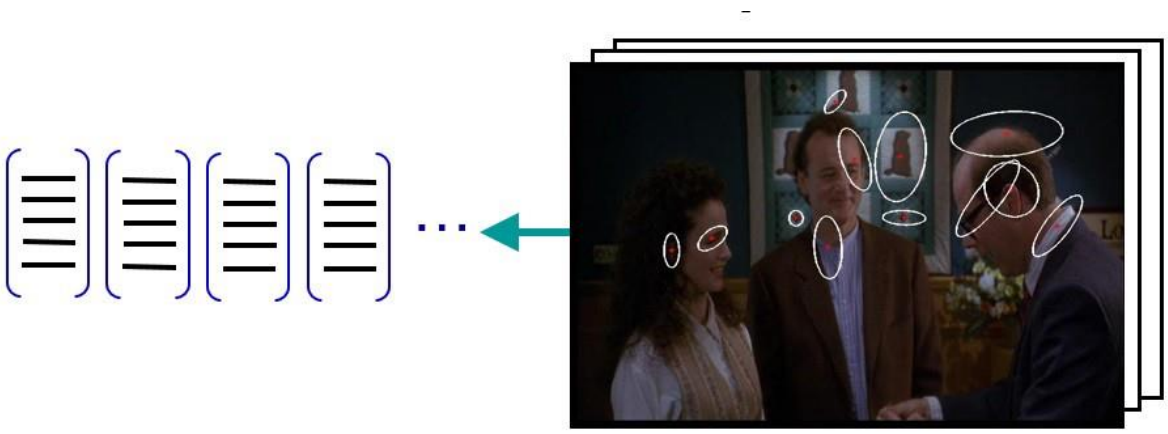
BoW – Bag of Words

Ogólną ideą worka z wizualnymi słowami (BOVW) jest przedstawienie obrazu jako zestawu cech. Cechy składają się z punktów kluczowych i deskryptorów. Punkty kluczowe są "wyróżniającymi się" punktami w obrazie, więc bez względu na to, czy obraz jest obrócony, zmniejszony czy rozwinięty, jego punkty kluczowe będą zawsze takie same. A deskryptor jest opisem punktu kluczowego. Używamy punktów kluczowych i deskryptorów do konstruowania słowników i reprezentujemy każdy obraz jako histogram częstotliwości cech, które są w obrazie. Na podstawie histogramu częstotliwości, później możemy znaleźć inne podobne obrazy lub przewidzieć kategorię obrazu.

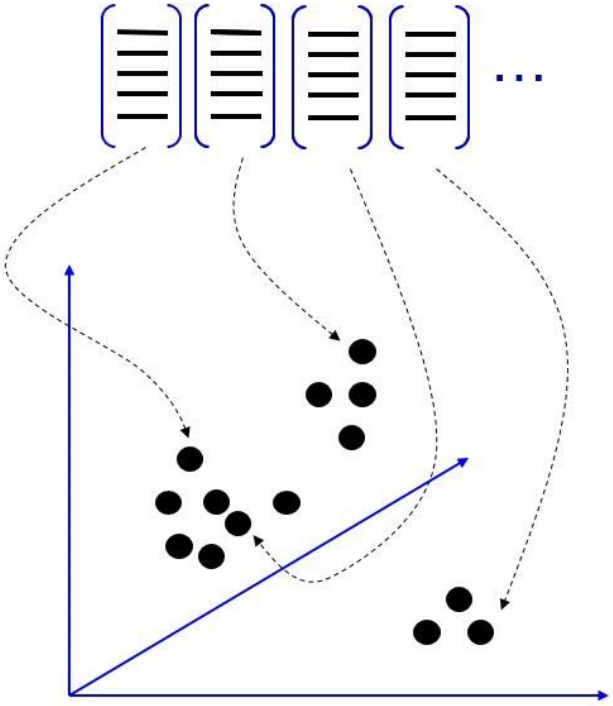


Jak zbudować torbę z wizualnymi słowami (BOVW)?

Wykrywamy cechy, wyodrębniamy deskryptory z każdego obrazu w zbiorze danych i budujemy słownik wizualny. Wykrywanie cech i wyodrębnianie deskryptorów w obrazie może odbywać się za pomocą algorytmów wyodrębniania cech (np. SIFT, KAZE, itp.).



Następnie tworzymy klastry z deskryptorów (możemy użyć K-Means, DBSCAN lub innego algorytmu klastrowania). Środek każdego klastra będzie służył jako słownik wizualny.



Na koniec, dla każdego obrazu, wykonujemy histogram częstotliwości ze słowników i częstotliwości słowników na obrazie. Te histogramy są naszym workiem z wizualnymi słowami (BOVW).

Sieci neuronowe

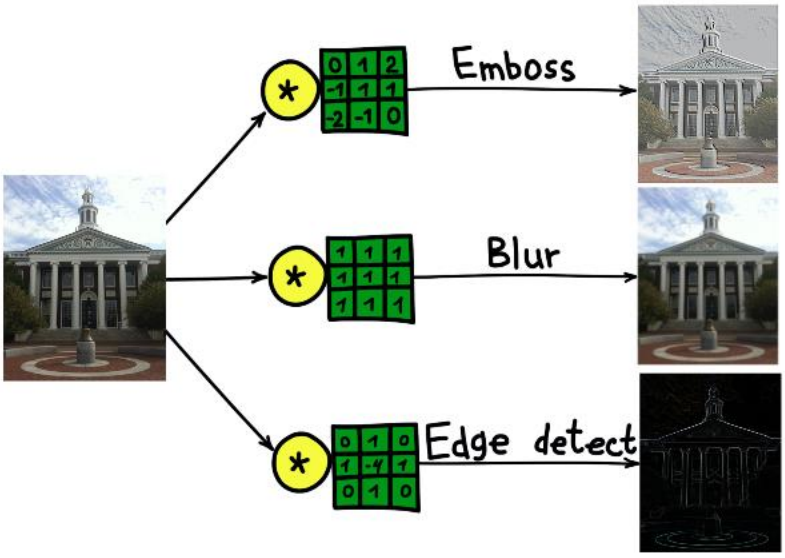
Z biegiem czasu zauważono, że większość technik generowania cech można uogólnić za pomocą jądra (filtrów) - małych macierzy (zwykle 5 × 5), które są splotami oryginalnych obrazów. Konwolucję można traktować jako sekwencyjny proces dwuetapowy:

1. Przejść przez cały oryginalny obraz z tym samym stałym jądrem.
2. Na każdym kroku obliczyć iloczyn skalarny jądra i oryginalnego obrazu w punkcie bieżącej lokalizacji jądra.

Wynik splotu obrazu i jądra nazywany jest mapą cech.

Prostym przykładem transformacji, której można dokonać za pomocą filtrów, jest rozmycie obrazu. Weźmy filtr składający się ze wszystkich jednostek. Oblicza średnią z sąsiedztwa zdefiniowanego przez filtr. W tym

przypadku sąsiedztwo jest obszarem kwadratowym, ale może mieć kształt krzyża lub cokolwiek innego. Uśrednianie prowadzi do utraty informacji o dokładnym położeniu obiektów, a tym samym do zatarcia całego obrazu. Podobne intuicyjne wyjaśnienie można podać dla każdego ręcznie utworzonego filtru.



Konwolucyjne podejście do klasyfikacji obrazów ma dwie istotne wady:

- Proces wieloetapowy zamiast sekwencji od końca do końca (end-to-end sequence).
- Filtry są świetnym narzędziem uogólniającym, ale są to stałe matryce. Jak dobrać wagi filtrów?

Na szczęście wynaleziono filtry, które da się trenować. Stanowią one podstawową zasadę działania konwolucyjnych sieci neuronowych. Zasada ta jest prosta: przeszkolimy filtry zastosowane do opisu obrazów tak, aby jak najlepiej spełniały one swoją rolę.

Konwolucyjne sieci neuronowe rozwiązują od razu dwa problemy: nie ma potrzeby wcześniejszego definiowania filtrów, a proces uczenia się przebiega od końca do końca. Typowa architektura konwolucyjnej sieci neuronowej składa się z następujących części:

- Warstwy splotowe
- Warstwy podpróbkowania
- Gęste (w pełni połączone) warstwy

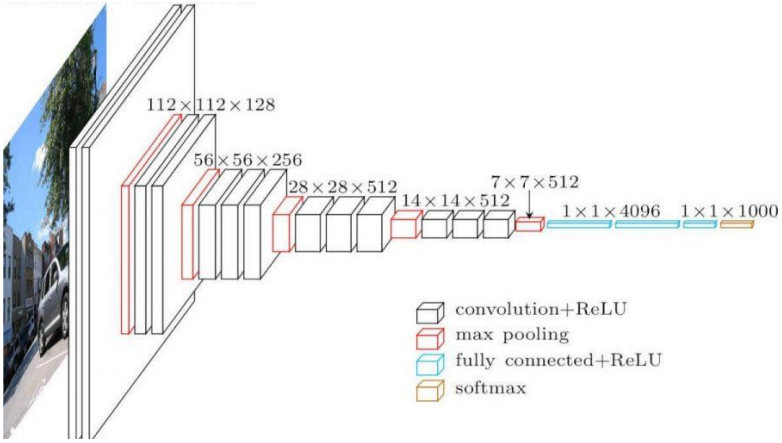
W warstwach splotowych neurony są połączone nie zbyt gęsto, a zadaniem tych warstw jest zbudowanie map cech, co pozwala na wyróżnienie znaczących cech obrazu. Przy czym stopień ich abstrakcji rośnie wraz z głębokością sieci. Może to być np. obecność pionowych/poziomych linii na pierwszych warstwach i obecność psa/kota/osoby na ostatnich.

Podpróbkowanie można traktować jako inny typ jądra. Służy ono trzem głównym celom: zmniejszeniu wymiaru przestrzennego (zmniejszenie liczby parametrów), wzrostu strefy podatności (ze względu na podpróbkowanie neuronów w kolejnych warstwach, gromadzono jest więcej kroków sygnału wejściowego) oraz niezmienności translacyjnej do małych nieciągłości w położeniu szablonów w sygnale wejściowym.

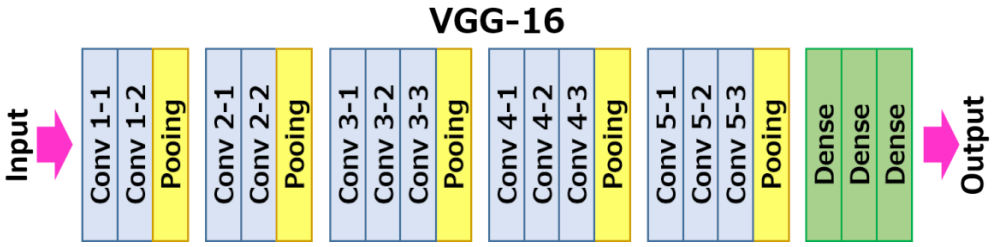
Ostatnim krokiem jest klasyfikacja obrazu wejściowego na podstawie wykrytych cech. W konwolucyjnych sieciach neuronowych robią to gęste warstwy na górze sieci. Ta część sieci nazywa się *klasyfikacją* . Może zawierać kilka warstw jedna na drugiej z pełną łącznością, ale zwykle kończy się warstwą aktywowaną przez wielozmienną funkcję aktywacji logistycznej, w której liczba bloków jest równa liczbie klas. Wynikiem tej warstwy jest rozkład prawdopodobieństwa na klasy dla obiektu wejściowego. Obraz można teraz sklasyfikować, wybierając najbardziej prawdopodobną klasę.

VGG16

VGG16 jest modelem sieci neuronowej konwulsyjnej. Model osiąga dokładność 92,7% - top 5, po przetestowaniu w sieci ImageNet w zadaniu rozpoznawania obiektów na obrazie. Ten zbiór danych składa się z ponad 14 milionów zdjęć należących do 1000 klas.



VGG16 jest jednym z najbardziej znanych modeli wysłanych na ILSVRC-2014. Jest to ulepszona wersja AlexNet, która zastąpiła duże filtry (rozmiar 11 i 5 w pierwszej i drugiej warstwie wiązki, odpowiednio) kilkoma następującymi po sobie filtrami 3x3. Sieć VGG16 była przez kilka tygodni szkolona przy użyciu kart graficznych NVIDIA TITAN BLACK.



Na wejściu warstwy conv1 podawane są obrazy RGB o rozmiarach 224x224. Obrazy przechodzą następnie przez stos warstw conv1, który wykorzystuje filtry z bardzo małym polem chłonnym 3x3 (który jest najmniejszym rozmiarem, aby zorientować się, gdzie znajduje się prawy/lewy, góra/dół, środek).

Jedna z konfiguracji wykorzystuje filtr 1x1 zwojowy, który może być przedstawiony jako liniowa transformacja kanałów wejściowych (z późniejszą nieliniowością). Krok zwijania jest ustalony na wartość 1 piksela. Dodatek przestrzenny (wypełnienie) wejścia warstwy zwinięcia jest tak dobrany, że rozdzielczość przestrzenna jest zachowana po zwinięciu, tzn. dodatek jest równy 1 dla 3x3 warstw zwinięcia. Pudding przestrzenny wykonywany jest za pomocą pięciu warstw o maksymalnym udźwigu, które następują po jednej z warstw wiązki (nie wszystkie warstwy wiązki mają kolejne warstwy o maksymalnym udźwigu). Operacja maksymalizacji jest wykonywana na oknie 2x2 pikseli w kroku 2.

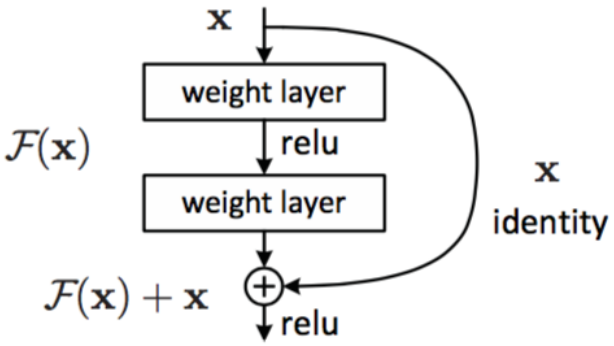
Po stosie warstw zwiniętych (które mają różną głębokość w różnych architekturach) znajdują się trzy w pełni połączone warstwy: dwie pierwsze mają 4096 kanałów, trzecia - 1000 kanałów (tak jak w konkursie ILSVRC wymagana jest klasyfikacja obiektów na 1000 kategorii, dlatego jeden kanał odpowiada klasie). Ostatnia z nich to warstwa soft-max. Konfiguracja w pełni połączonych warstw jest taka sama we wszystkich sieciach neuronowych.

Niestety, sieć VGG ma dwie poważne wady:

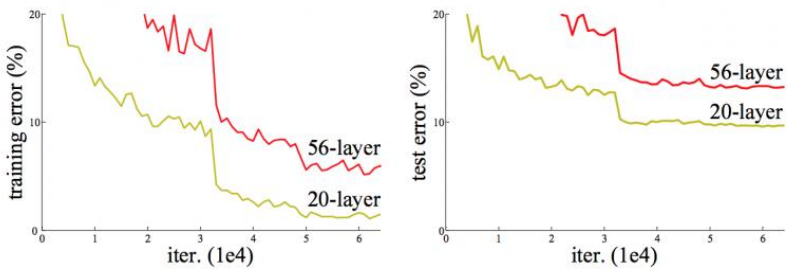
- Bardzo wolne tempo nauki.
- Sama architektura sieciowa waży za dużo (są problemy z dyskami i pasmem)

- Ze względu na głębokość i liczbę w pełni połączonych węzłów, VGG16 waży ponad 533 MB. To sprawia, że proces wdrażania VGG jest żmudny. Chociaż VGG16 jest używany do rozwiązywania wielu problemów klasyfikacyjnych z sieciami neuronowymi, preferowane są mniejsze architektury (SqueezeNet, GoogLeNet i inne). Pomimo swoich wad, architektura ta jest doskonałym elementem konstrukcyjnym dla szkoleń, ponieważ jest łatwa do wdrożenia.

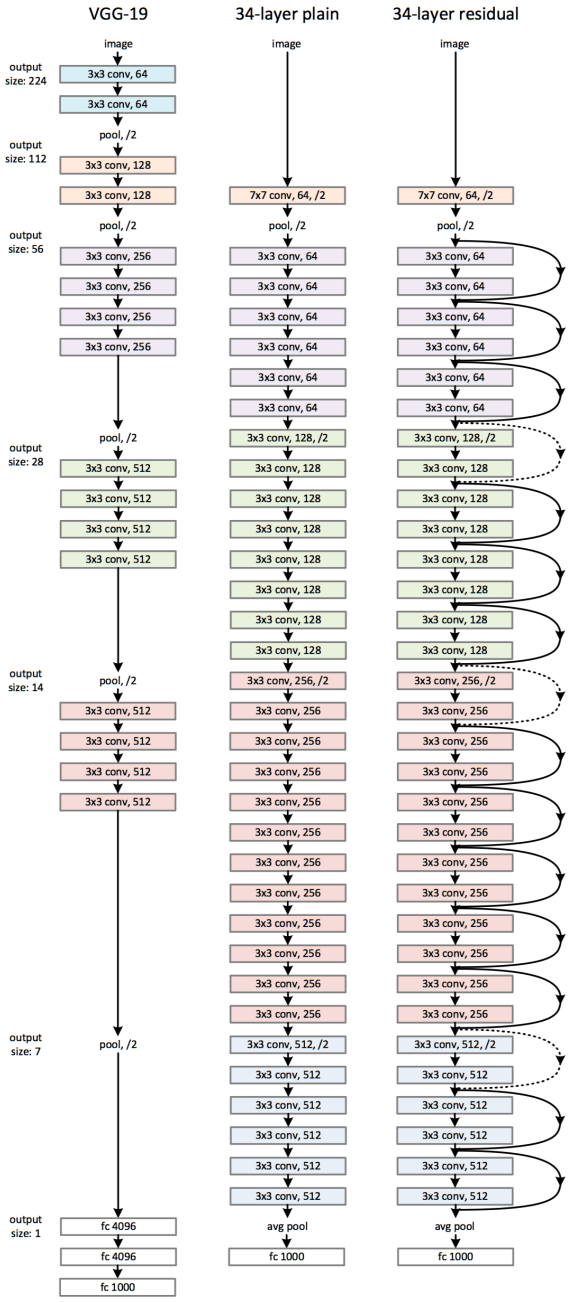
Resnet



ResNet jest skrótem od Residual Network. Głęboko chrupiące sieci neuronowe przekroczyły ludzki poziom klasyfikacji obrazu w 2015 roku. Głębokie sieci wydobywają cechy niskiego, średniego i wysokiego poziomu w sposób poprzeczny, wielowarstwowy, natomiast zwiększenie liczby ułożonych warstw może wzbogacić "poziomy" cech. Warstwa ułożona w stos jest kluczowa, spójrz na wynik sieci ImageNet:



Kiedy głębsza sieć zaczyna zapadać, problem polega na tym, że wraz ze wzrostem głębokości sieci najpierw zwiększa się dokładność, a następnie gwałtownie się pogarsza. Zmniejszona dokładność uczenia się pokazuje, że nie wszystkie sieci są łatwe do zoptymalizowania.



Simple Network: Proste linie bazowe są głównie inspirowane filozofią sieci VGG. Warstwy wiązki mają głównie filtry 3×3 i działają według dwóch prostych zasad:

Dla tej samej wyjściowej mapy obiektów, warstwy mają taką samą liczbę filtrów;

Jeśli rozmiar mapy obiektu zostanie zmniejszony o połowę, liczba filtrów zostanie podwojona, aby zachować czasową złożoność każdej z warstw.

Warto zauważyć, że model ResNet ma mniej filtrów i jest mniej skomplikowany niż sieci VGG.

ResNet: w oparciu o prostą sieć opisaną powyżej, dodano szybkie połączenie, które zamienia sieć w jej pozostałą wersję. Szybkie połączenia identyfikacyjne F (x {W} + x) mogą być używane bezpośrednio, gdy wejście i wyjście mają te same. Gdy wymiary są zwiększone, rozważa on dwie opcje:

- Szybkie połączenie wykonuje porównanie ID z dodatkowymi zerami dodanymi w celu zwiększenia wymiarowości. Opcja ta nie wprowadza żadnych dodatkowych parametrów.
- Projekcja Quick Connect w F (x {W} + x) jest używana do mapowania wymiarowego (wykonywanego z 1×1 zwojami).

Dla każdej z tych opcji, jeśli szybkie połączenia są wykonywane na dwuwymiarowych mapach obiektów, są one wykonywane w kroku 2.

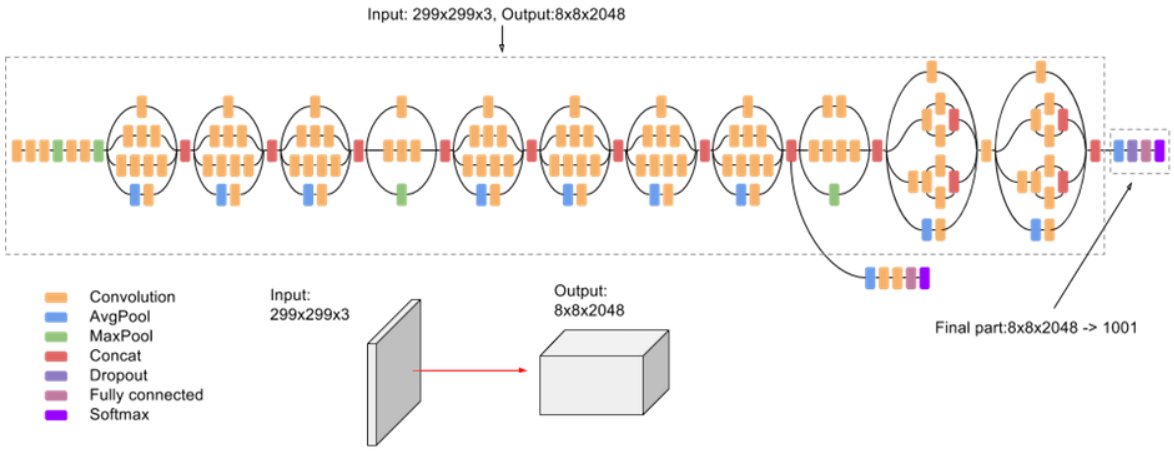
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Inception-v3

Inception v3 jest szeroko stosowanym modelem rozpoznawania obrazu, który w zbiorze danych ImageNet osiągnął dokładność większą niż 78,1%. Model ten jest zwieńczeniem wielu pomysłów opracowanych przez wielu badaczy na przestrzeni lat. Jest on oparty na oryginalnym papierze: "Rethinking the Inception Architecture for Computer Vision" autorstwa Szegedy, et. al.

Sam model składa się z symetrycznych i asymetrycznych elementów konstrukcyjnych, w tym: zwojów, basenów średnich, basenów maksymalnych, betonów, zrzutów i w pełni połączonych warstw. Batchnorm jest szeroko stosowany w całym modelu i stosowany do wejść aktywacyjnych. Strata jest obliczana za pomocą programu Softmax.

Schemat wysokiego poziomu modelu został przedstawiony poniżej:

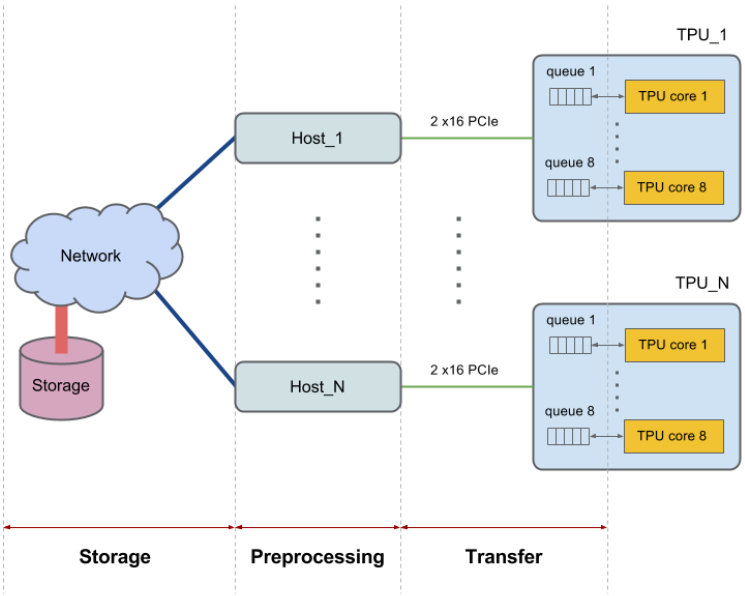


Wersja TPU Inception v3 jest napisana przy użyciu TPUEstimatora, API zaprojektowanego w celu ułatwienia rozwoju, tak aby można było skupić się na samych modelach, a nie na szczegółach podstawowego sprzętu. API wykonuje większość niskopoziomowych prac grunge'owych niezbędnych do uruchamiania modeli na TPU za kulisami, jednocześnie automatyzując wspólne funkcje, takie jak zapisywanie i przywracanie punktów kontrolnych.

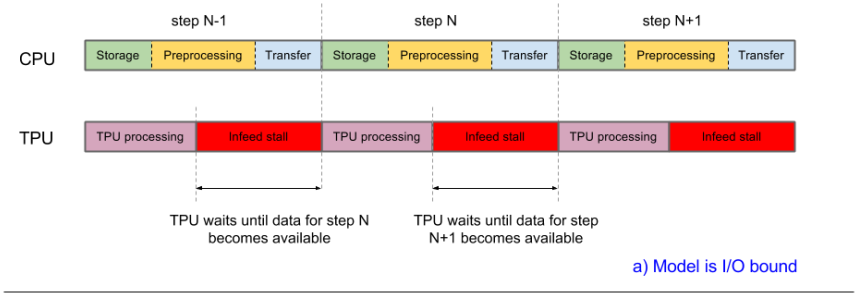
API Estymator wymusza oddzielenie części modelu od części wejściowej kodu. Musisz zdefiniować funkcje model_fn i input_fn, odpowiadające odpowiednio etapom definiowania modelu i potoku wejściowego / wstępnego przetwarzania wykresu TensorFlow.

Każde urządzenie TPU Cloud posiada 8 rdzeni i jest podłączone do hosta (CPU). Większe plasterki mają wiele hostów. Inne większe konfiguracje współdziałają z wieloma hostami. Na przykład, v2-256 komunikuje się z 16 hostami.

Hosty pobierają dane z systemu plików lub pamięci lokalnej, robią wszystko, co jest wymagane do wstępnego przetworzenia danych, a następnie przesyłają wstępnie przetworzone dane do rdzeni TPU. Te trzy fazy obsługi danych wykonywane przez hosta traktujemy indywidualnie i określamy je jako: 1) Przechowywanie, 2) Przetwarzanie wstępne, 3) Transfer. Wysokopoziomowy obraz schematu przedstawia poniższy rysunek:



Aby uzyskać dobre wyniki, system powinien być zrównoważony. Niezależnie od tego, ile czasu procesor hosta spędza na pobieraniu obrazów, dekodowaniu ich i odpowiednim przetwarzaniu wstępnym, najlepiej byłoby, gdyby czas ten był nieco krótszy lub mniej więcej taki sam jak czas spędzony przez TPU na wykonywaniu obliczeń. Jeśli procesor hosta zajmie więcej czasu niż TPU na zakończenie trzech faz przetwarzania danych, wówczas wykonanie będzie związane z hostem. (Uwaga: ponieważ procesory TPU są tak szybkie, może to być nieuniknione w przypadku niektórych bardzo prostych modeli). Oba przypadki zostały przedstawione na poniższym schemacie.



FIX RES

Została zaprojektowana przez Facebook AI Team oraz EfficientNet. Jej nazwa jest skrótem od Fix Resolution i ona stara się utrzymywać stały rozmiar zarówno obszaru klasyfikacji dla trenowania, jak i testowanego wycinku zdjęcia.

Trenowanie:

Dopóki zespół badawczy AI Facebooka nie zaproponował techniki FixRes, najnowocześniejszym sposobem było wyodrębnienie losowego kwadratu pikseli z obrazu. To było używane jako obszar klasyfikacji na czas szkolenia. Obraz został następnie zmieniony, aby uzyskać obraz o stałym rozmiarze (= przycięcie). Zostało to następnie przesłane do konwolucyjnej sieci neuronowej.

Zmiana rozmiaru obszaru klasyfikacji w obrazie wejściowym wpływa na rozkład rozmiaru obiektu przekazanego konwolucyjnej sieci neuronowej. Obiekt ma rozmiar ***r*** *x* ***r*** w obrazie wejściowym. Jeśli obszar klasyfikacji jest przeskalowany, jest on zmieniany razy ***s***, a więc rozmiar obiektu wynosi ***rs*** *x* ***rs***.

Do wzmocnienia używa się RandomResizedCrop. Obraz wejściowy ma rozmiar ***H*** *x* ***W***, z którego wybiera się losowo obszar klasyfikacji, który jest następnie zmieniany na rozmiar zbioru:

$$K_{train}\ x\ K_{train}$$

Skalowanie obrazu wejściowego (***H*** *x* ***W***) do przycięcia wyjściowego można wyrazić następującym współczynnikiem:

$$s = \frac{\sqrt{K_{train}\ x\ K_{train}}}{\sqrt{H_{RoC}\ x\ W_{RoC}}} = \frac{1}{\sigma}\ x\ \frac{K_{train}}{\sqrt{HW}}$$

Testowanie:

W czasie testu obszar klasyfikacji jest często wyśrodkowany na obrazie, co daje tak zwany środkowy kadr. Obojga przycięcia, ta z czasu trenowania i z czasu testowania, mają ten sam rozmiar, ale pochodzą z innej części obrazu. Często prowadzi to do tendencyjności w dystrybucji konwolucyjnej sieci neuronowej.

Jak opisano wcześniej, wzmocnienie testowe nie jest tym samym, co zwiększenie czasu treningu. Przycięcie ma więc rozmiar K_{test} *x* K_{test}.

Biorąc pod uwagę założenie, że obraz wejściowy jest kwadratem (H = W), współczynnik skalowania dla wzmocnienia testu można wyrazić jako **s = K_{test}/sqrt(HW)**.

Do czasu opracowania FixRes przetwarzanie wstępne dla czasu testowania i szkolenia było oddzielone od siebie, co prowadziło do błędów. Zespół AI Facebooka próbował znaleźć rozwiązanie, które jednocześnie wykonuje wstępne przetwarzanie i testowanie w sposób zsynchronizowany, i jest nim FixRes.

Technika FixRes przyjmuje podejście albo-albo. Zmniejsza rozdzielczość czasu treningu i utrzymuje rozmiar uprawy testowej lub zwiększa rozdzielczość dla testowania i utrzymuje rozmiar uprawy szkoleniowej. Celem jest pobranie obiektu o tej samej wielkości, aby zmniejszyć niezmienniczość skali w konwolucyjnej sieci neuronowej. Prowadza to do dwóch skutków w sposobie dostarczania danych do CNN:

- Rozmiar obiektu na obrazie jest zmieniany przez FixRes Scaling.
- Stosowanie różnych rozmiarów upraw ma wpływ na to, jak i kiedy aktywowane są neurony.

Wnioski:

Przebadaliśmy metody wyróżniania cech oraz kilka najpopularniejszych podejść do klasyfikacji obrazów. Dzisiaj klasyczne metody odeszły z powodu niższej dokładności i zostały zamienione na sieci neuronowe. Istnieje mnóstwo architektur konwolucyjnych sieci neuronowych do klasyfikacji i rozpoznawania obrazów. W tym raporcie zostały omówione takie architektury jak VGG-16, ResNet, Inception-v3 oraz FixRes. Najbardziej rozwijaną jest teraz architektura Inception od Google. Istnieje już wersja v4, ale dla tego badania byłoby to zbyt dużą ilością informacji.

Źródła:

https://arxiv.org/pdf/1906.06423v3.pdf

https://cloud.google.com/tpu/docs/inception-v3-advanced

https://habr.com/ru/company/intel/blog/415811/

https://neurohive.io/ru/vidy-nejrosetej/resnet-34-50-101/

https://neurohive.io/ru/vidy-nejrosetej/vgg16-model/

https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb

https://towardsdatascience.com/state-of-the-art-image-classification-algorithm-fixefficientnet-l2-98b93deeb04c

https://www.learnopencv.com/histogram-of-oriented-gradients/

https://www.pyimagesearch.com/2014/01/22/clever-girl-a-guide-to-utilizing-color-histograms-for-computer-vision-and-image-search-engines/

https://www.pyimagesearch.com/2016/08/08/k-nn-classifier-for-image-classification/

Podział zadań na I etap:

Aliaksei	**Daniil**	**Aliaksandr**
Bag Of Words	kNN	Support Vector Machine
Harris Corner Detection	Color Gradient Histogram	Histogram of Oriented Gradients

Terminy:

Etap 1: 03.11 – 25pkt.

Etap 2: 24.11 – 50pkt.

Etap 3: 08.12 – 25pkt.

Sprawozdania z każdego etapu.