POLITECHNIKA KRAKOWSKA

Program optymalizujący sieci konwolucjne w celu uzyskania modelu z najmniejszym błędem uczenia

Projekt z przedmiotu

Automaty, Języki i Obliczenia

Filip Kostkiewicz

130529

Kraków 2021

Spis treści

[Wstęp 3](#_Toc88814546)

[Cel projektu 3](#_Toc88814547)

[Technologia 3](#_Toc88814548)

[Sieci konwolucjne 3](#_Toc88814549)

[Skąd pomysł na sieci konwolucjne 3](#_Toc88814550)

[Główne elementy sieci 4](#_Toc88814551)

[Schemat działania sieci 4](#_Toc88814552)

[Opis systemu 5](#_Toc88814553)

[Schemat działania programu 5](#_Toc88814554)

[Wymagania funkcjonalne 5](#_Toc88814555)

[Dane wejściowe 6](#_Toc88814556)

[Opis działania elementów systemu 6](#_Toc88814557)

[Okno danych ogólnych 6](#_Toc88814558)

[Okno wprowadzania do ustawiania zmiennych warstw 6](#_Toc88814559)

[Okno z wynikami 7](#_Toc88814560)

[Testy 7](#_Toc88814561)

[Wykrywanie niepoprawnych danych (poziom użytkownika) 7](#_Toc88814562)

[Wykrywanie niepoprawnych danych (poziom programisty) 7](#_Toc88814563)

[Wnioski 7](#_Toc88814564)

[Bibliografia 7](#_Toc88814565)

Wstęp

Moim tematem pracy inżynierskiej jest „Rozpoznawanie emocji w czasie rzeczywistym na podstawie twarzy”, program ten rozpoznaje 7 emocji podstawowych, do uczenia się wzorców wykorzystuje sieci konwolucjne. Sieci te są rozszerzeniem sieci neuronowej, oraz posiadają one większą liczbę parametrów wpływających na proces uczenia.

Chęć uzyskania jak najlepszego modelu, posiadającego dużą liczbę parametrów sprawiła zmianę podejścia do stawianego problemu, zamiast mozolnie przeszukiwać ręcznie, przeróżne kombinacje, najlepiej byłoby stworzenie programu który wykona to za mnie. Użytkownik programu podawałby jedynie przedziały w których dany program powinien szukać.

Kolejnym aspektem który miał wpływ na stworzenie danego programu jest możliwość zawarcia go w mojej pracy inżynierskiej.

Cel projektu

Głównym celem projektu jest napisanie programu który na podstawie podanych wartości (parametrów warstw sieci konwolucjnej) przez użytkownika, sprawdzi wszystkie możliwe kombinacje, zwracając parametry modelu który uzyskał najmilejszy błąd uczenia. Przez co program ten mogę zastosować do uzyskania najlepszego modelu uczącego, w mojej pracy inżynierskiej.

Kolejnym celem projektu jest uzyskanie programu który pomoże użytkownikowi w wyborze najlepszych parametrów sieci konwolucjnej, zamiast mozolnego sprawdzania wielu parametrów. Program ten można zastosować nie tylko do mojej pracy, lecz jest uniwersalny gdyż można go zastosować do uczenia każdej sieci konwolucjnej.

Technologia

Od wielu lat język programowania Python przoduje w programach do uczenia maszynowego oraz posiada on bardzo dobre biblioteki do trenowania takie jak tensorflow i keras. Zarówno posiada on bardzo intuicyjna i łatwą do obsługi bibliotekę tkiner, służąca do wyświetlania okien graficznych

Sieci konwolucjne

Pisząc program do optymalizacji sieci konwolucjnej warto wpierw zagłębić się nad działaniem tej sieci.

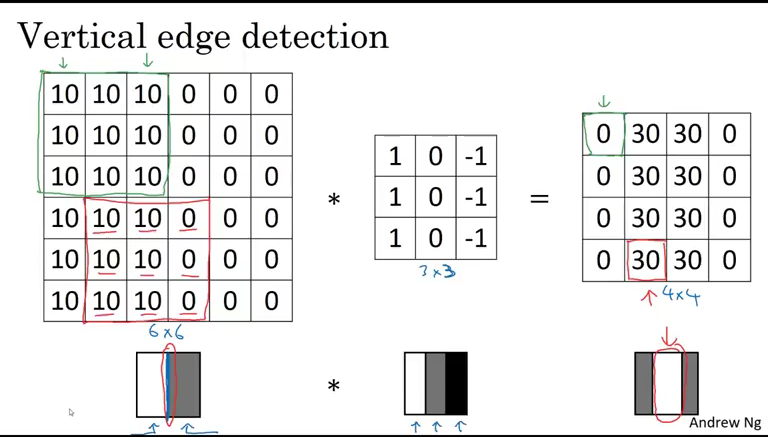
Skąd pomysł na sieci konwolucjne

Sieci konwolucjne są stosowane od niedawna, i są kolejną generacją sieci neuronowej z tą różnicą ze sieci neuronowe uczyły się globalnych cech a sieci konwolucjne lokalny cech. Sieci te stają się popularne ponieważ sieci neuronowe w dużej mierze uczył się na różnego rodzaju zdjęć. Rozwój techniki aparatów i kamer umożliwił tworzenie zdjęć z coraz większą rozdzielczością np. 4k, spowodowało to że wektor wejściowy sieci neuronowej miałby długość 8294400 neuronów (3840x2160 sze/wys zdjęcia), jeśli kolejna warstwa miała by tylko 100 neuronów, dało by to 829,4 miliona wag, jest to ogromna liczba do informacji zapamiętania dla komputera. To spowodowało zainteresowanie sieciami konwolucjnymi gdyż stosując przeróżne filtry na obrazie, zagęszcza cechy obrazu, zmniejszając rozmiar, uzyskuje przez to cechy lokalne dzięki którym sieć potrafi dużo skuteczniej się uczyć niż sieci neuronowe.

Główne elementy sieci

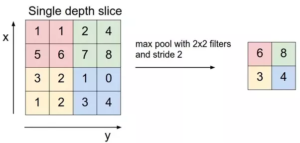
**Normalizacja wartości koloru** – zdjęcia jpg lub png posługują się modelem przeszczepi barw RGB zapisuje on 3 podstawowe kolory każdy w przedziale 0-255. Dla sprawniejszego funkcjonowania sieci warto przeskalować wszystkie wartości wejściowe przez 255, ponieważ sieci lepiej działają na wartościach z przedziału 0-1.

**Warstwa konwolucjna(splotowa)** – aby uzyskać cechy z obrazu, filtry przechodzą przez każdy piksel obrazu, stosując różne filtry małego rozmiaru 3x3, 5x5. Filery te trwożą nowy obraz z wyodrębnionymi różnymi cechy np. linie poziome lub skośne.



Przykład konwolucji[1]

**Maxpooling** – służy do zmniejszenia otrzymanych obrazów przez filtr z zagęszczeniem cech wyznaczonych przez filtr, poprzez zamianę obszaru nxm piksele na jeden piksel o największej wartości. Operacja ta działa w ten sposób że zamiast, miejscowo wiele cech tego samego typu zamieniana jest na jeden pksel, zyskujemy tym więcej miejsca i czasu.

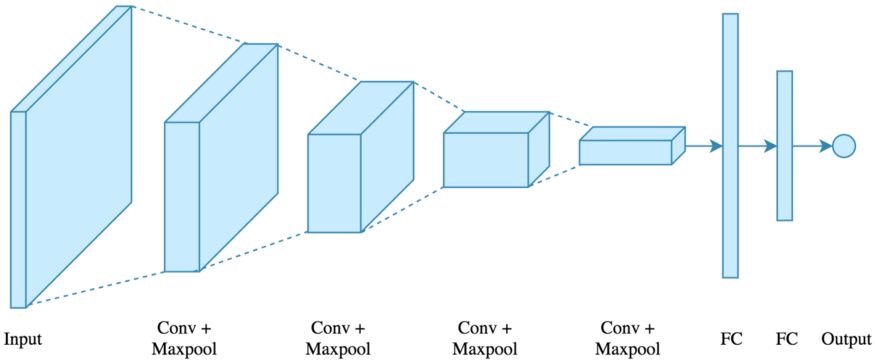


Przykład maxpooling[2]

Schemat działania sieci

Zanim zdjęcie zostanie wykorzystane do uczenia normalizuje się je. Siec konwolucjna działa w sposób następujący:

1. Jest tworzone m obrazów(splotów) każdy powstał przez inny filtr.
2. Na każdym splocie wykonywany jest maxpooling.
3. Następnie podpunkty 1 i 2 wykonujemy kilkukrotnie, za każdym razem obliczając nowe sploty na podstawie splotów z poprzedniej warstwy, oraz zwiększaniem liczbę splotów co serie.
4. Podpunkt 3 wykonujemy dopóki maxpooling nie zwróci obrazu o wymiarach ok 10x10, wtedy wszystkie sploty ostatniej warstwy są rozwijane do postaci wektora i łączone z innymi wektorami w jeden długi wektor.
5. Otrzymany wektor jest warstwą wejściową dla sieci neuronowej.



Przykład sieci konwolucjnej[3]

Conv – convolution (konwolucja)

FC -  fully connected layers (warstwy neuronowe tzw. warstwy gęste)

Opis systemu

Schemat działania programu

* Wprowadzanie danych ogólnych
* Sprawdzanie poprawności wprowadzonych danych ogólnych
* Wprowadzanie zmiennych dla każdej warstwy
* Sprawdzanie poprawności wprowadzonych zmiennych dla warstw
* Sprawdzanie wszystkich możliwości
* Wyświetlenie uzyskanych wyników

Wymagania funkcjonalne

Użytkownik ma mieć możliwość modyfikowania każdej warstwy osobno wpisując wartości oddzielone znakiem ‘,’ w wyznaczonych polach, lub zaznaczając odpowiednie opcje do zaznaczania.

Aplikacja powinna mieć osobne pole na wprowadzanie wartości takich jak ścieżka do pliku z danymi, sposób sprawdzania sieci.

Aby uruchomić funkcje sprawdzającej należy przycisnąć przycisk start. Aby funkcja ta mgła zostać uruchomiona niezbędne jest poprawne wypełnienie wszystkich podanych pul, w przypadku wykrycia błędu, nad przyciskiem start wyświetlany jest czerwony napis o wykrytym błędzie.

Wyniki wyświetlają się w nowym oknie, w układzie kolumnowym, najlepsze wyniki na gurze najgorsze na dole, każdy pojedynczy wiersz kolumny zawiera od lewej: numer test, uzyskana wartość i przycisk wyświetlający parametry danej warstwy.

Dane wejściowe

Program do obliczania błędu potrzebuje danych uczących i walidacyjnych.

**Sieci neuronowe** wymagają folderu zawierającego cztery pliki z rozszerzeniem csv, oznaczone odpowiednią nazwą train\_x – zbiór uczący wejściowy, train\_y – zbiór uczący wyjście, val\_x – zbiór walidacyjny wejściowy, val\_y – zbiór walidacyjny wyjście. Zbiory wejściowe zapisane są w sposób jeden wiersz to pierwsza warstwa. Zbiorach wyjściowych stany są zapisane jako liczby naturalne oznaczające odpowiednie emocje.

**Sieci konwolucjne** wymagają ścisłego ułożenia folderów i plików. Folder główny posiada dwa foldery „test” i „val” w obu tych folderach znajduje się ta sama liczba folderów tak samo nazwanych (nazwy klas emocji), oraz każdy z tych folderów musi zawierać jedynie pliki png.

Opis działania elementów systemu

Okno danych ogólnych

Jest to pierwsze okno graficzne wyświetlające się po uruchomieniu programu. Składa się 6 elementów, idąc od góry w dół: tytuł, ramka wyboru sieci, ramka wyboru funkcji sprawdzającej wszystkie kombinacje, ramka wyboru ścieżki do danych uczących, linia komunikatu o błędzie, przycisk zatwierdzający i uruchamiający kolejne okno.

Przyciśnięci przycisku powoduje sprawdzenie czy wszystkie ramki zostały wypełnione poprawnymi danymi:

* Ramka sieci: Czy została zaznaczona siec
* Ramka sprawdzająca: Czy została wybrana zaznaczona opcja i czy podana wartość tej opcji przez użytkownika jest liczba naturalna.
* Ramka ścieżki do pliku: ścieżka prowadzi do folderu, czy struktura w folderze jest poprawna dla konsolacji: dwa foldery, posiadające tak samo nazwane foldery wypełnione tylko zdjęciami, a dla sieci neuronowej cztery pliki z rozszerzeniem csv.

Gdy zostanie wykryty błąd, jest on wyświetlany w linii błędów, jeśli nie jest usuwana cała zawartości okna wyświetlając na niej okno do wprowadzania zmiennych warstwy uczącej.

Okno wprowadzania do ustawiania zmiennych warstw

Okno to służy do określenia parametrów jakie program ma sprawdzać podczas uczenia sieci. Jeśli w oknie poprzednim została wybrana opcja sieci neuronowej to pola do modyfikacji sieci konwolucjnej się nie pokażą. Obie sieci dosiadają przycisk do dodawania ramki reprezentującą jedną warstwę sieci, przyciski znajdują się zawsze pod ostatnią warstwą danej sieci. Kolejność ramek jest odzwierciedlana w warstwie uczącej, górna ramka jest pierwszą warstwą w sieci a ostatnia jest przedostatnia warstwą w sieci, ostatnia warstwa w sieci jest tworzona przez program na podstawie liczby elementów w zbiorze uczącym. Każda ramka posiada przycisk do usuwania danej warstwy, oraz pola do wpisywania paramentów warstwy.

Wciśniecie przycisku start powoduje sprawdzanie poprawności wszystkich podanych opcji:

* Czy wszystkie pola zostały wypełnione
* Czy wpisana liczba jest liczbą naturalna lub z przedział u (0, 1) w zależności od opcji.

Jeśli zostanie wykryty błąd nad przyciskiem wyświetla się jego treść, jeśli nie wykryto błędu usuwane są wszystkie elementy okna i uruchamiany jest okno z wynikami.

Okno z wynikami

Okno to służy do wyświetlania uzyskanych wyników. Program na podstawie podanych warstw tworzy tablice z wszystkimi kombinacjami podanymi przez użytkownika, następnie uruchamiany jest watek który sprawdza wszystkie kombinacje zwracając uzyskany błąd. Wątek jest niezbędny aby program mógł na bieżąca wyświetlać wyniki na ekranie. Wyniki są sortowane rosnąco i wyświetlane w kolumnie. Przyciśniecie przycisku „Info” otwiera osobne okno z informacjami o użytej sieci i jej parametrami. W przypadku błędu lub braku zdolności uczących sieci zamiast uzyskanego błędu wyświetlany jest napis „ERROR” oraz taki element jest podczas sortowania ustawiany na końcu listy. Związku z dość szybkim przyrostem wielu kombinacji i wyszukiwaniem najmniejszego błędy, program wyświetla jedynie 50 najlepszych wyników na ekranie.

Testy

Wykrywanie niepoprawnych danych (poziom użytkownika)

Okno początkowe





Okno wprowadzania danych do warstwy





Testowane funkcje wykazały wykrywanie niepoprawnych danych przez użytkownika.

Wykrywanie niepoprawnych danych (poziom programisty)

Klasa testująca Testy mająca cztery funkcje posiadające razem 62 testy jednostkowe sprawdza funkcje klas CheckData i FrameCheck, odpowiedzialne za sprawdzanie wprowadzonych danych przez użytkownika. Testy jednostkowe obu klas wykazały poprawne działanie testowanych funkcji.

Wnioski

Napisany program spełnił początkowe założone postawione na etapie planowania, chociaż przekroczył nieco on liczbę linki kodu szacowaną na etapie planowania. Test funkcji sprawdzającej wykazał w przypadku złożonych sieci czas obliczania jednej epoki znacznie się wydłuża, także liczba parametrów do sprawdzenia wydłuża znacznie działanie programu.

Bibliografia

1. [https://miroslawmamczur.pl/jak-dzialaja-konwolucyjne-sieci-neuronowe-cnn/](https://miroslawmamczur.pl/jak-dzialaja-konwolucyjne-sieci-neuronowe-cnn/%20(12.11.2021)) (12.11.2021)
2. <https://bfirst.tech/konwolucyjne-sieci-neuronowe/> (12.11.2021)
3. <https://medium.com/@kristian.wichmann/making-mobilemeteor-56440804fee0> (12.11.2021)