Tekst do prezentacji dotyczącej projektu zaliczeniowego

1. **Slajd wprowadzający**

2. **Część teoretyczna**

3. **Uczenie maszynowe**
   Uczenie maszynowe to używanie matematycznych modeli danych w celu ułatwienia komputerowi uczenia się bez bezpośrednich instrukcji. Jest ono traktowane jako podzbiór sztucznej inteligencji. Algorytmy używane w uczeniu maszynowym umożliwiają określanie wzorców w danych. Wzorce te są następnie używane do tworzenia modelu danych, który pozwala przewidywać. Dokładność wyników uczenia maszynowego zwiększa się wraz z upływem czasu i wzrostem ilości danych - podobnie jak u ludzi. Elastyczność uczenia maszynowego sprawia, że staje się ono świetnym wyborem w scenariuszach, w których dane lub charakter żądania bądź zadania stale się zmieniają, albo w sytuacjach, w których zakodowanie rozwiązania byłoby w praktyce niemożliwe.
   OBRAZ: Uczenie maszynowe.
   OBRAZ: Składniki uczenia komputera - *artificial inteligence* (sztuczna inteligencja), *machine learning* (uczenie maszynowe) i *deep learning* (uczenie głębokie).

4. **Biblioteka TensorFlow**
   TensorFlow to bardzo rozbudowana otwartoźródłowa biblioteka programistyczna do *machine learningu* od Google Brain Team. Cechuje się wysoką wydajnością i skalowalnością, choć uczy się wolniej w porównaniu do SciKit. Z drugiej strony, posiada ona wbudowane wsparcie do pracy z urządzeniami z układami GPU - do swojego działania może wykorzystywać zarówno karty graficzne, procesory (m.in. dla urządzeń mobilnych oraz systemów wbudowanych), jak i wyspecjalizowane mikroprocesory nazywane akceleratorami AI *(tensor processing unit)*.
   Elastyczna architektura TensorFlow umożliwia wdrożenie obliczeń na jednym lub większej liczbie procesorów (CPU) lub kart graficznych (GPU) na komputerze osobistym, serwerze itp. bez konieczności ponownego pisania kodu. Firma Google stworzyła także Tensor Processing Units (inaczej TPU), wyspecjalizowany układ scalony, zaprojektowany z myślą o rozwoju maszynowego uczenia się i stosowania TensorFlow. TPU zostały zaprojektowane do wykorzystywania i testowania modeli, a nie do ich uczenia się. Od lutego 2018 r. TPU są dostępne w wersji beta Google Cloud Platform.
   OBRAZ: Logo TensorFlow.
   OBRAZ: Struktura biblioteki.
   OBRAZ: *Machine learning*.

5. **Biblioteka TensorFlow**
   Biblioteka składa się z kilku modułów. W jej najniższej warstwie znajduje się rozproszony silnik wykonawczy *(distributed execution engine)*, który w celu podniesienia wydajności został zaimplementowany w języku programowania C++. Nad nią znajdują się frontendy napisane w kilku językach programowania m.in. w Pythonie oraz C++. Powyżej umieszczona została warstwa API, która zapewnia prostszy interfejs dla powszechnie używanych warstw w modelach głębokiego uczenia. Na następną warstwę składają się wysokopoziomowe API, m.in. Keras oraz Estimator API, które ułatwiają tworzenie modeli i ich ocenę. Ponad tym znajdują się

przygotowane przez twórców biblioteki z gotowymi modelami do użycia.
OBRAZ: Budowa modułowa biblioteki TensorFlow.

6. TensorFlow - cechy

Bibliotekę TensorFlow charakteryzuje:
Automatyczne różnicowanie - to proces automatycznego obliczania wektora gradientu modelu w odniesieniu do każdego z jego parametrów. Dzięki tej funkcji TensorFlow może automatycznie obliczać gradienty parametrów w modelu.
OBRAZ: Obliczenie gradientu parametrów w modelu.
Eager execution - TensorFlow zawiera tryb „Eager execution", co oznacza, że operacje są oceniane natychmiast, w przeciwieństwie do dodawania ich do grafu obliczeniowego, który jest wykonywany później. Kod wykonywany może być zbadany krok po kroku przez debugger, ponieważ dane są powiększane w każdym wierszu kodu, a nie później w postaci wykresu obliczeniowego.
Dostępność - zarówno w przypadku wykonywania szybkich, jak i grafowych, TensorFlow zapewnia interfejs API do dystrybucji obliczeń na wiele urządzeń z różnymi strategiami dystrybucji. Takie przetwarzanie rozproszone może często przyspieszyć wykonywanie uczenia i oceny modeli TensorFlow i jest powszechną praktyką w dziedzinie sztucznej inteligencji.
OBRAZ: Wysoka dostępność biblioteki TensorFlow na podstawie modułu Graph.
Wykorzystanie funkcji strat - aby trenować i oceniać modele, TensorFlow zapewnia zestaw funkcji strat (znanych również jako funkcje kosztów). Niektóre popularne przykłady obejmują błąd średniokwadratowy (MSE) i binarną entropię krzyżową (BCE). Te funkcje straty obliczają „błąd" lub „różnicę" między wynikiem modelu a wynikiem oczekiwanym (szerzej, to różnica między dwoma tensorami).
Metryki oceny - aby ocenić wydajność modeli uczenia maszynowego, TensorFlow udostępnia API do powszechnie używanych metryk. Przykłady obejmują różne metryki dokładności (binarne, kategoryczne, rzadkie kategoryczne) wraz z innymi metrykami, takimi jak precyzja.
TF.nn - TensorFlow.nn to moduł do wykonywania prymitywnych operacji sieci neuronowych na modelach.
Optymalizatory - TensorFlow oferuje zestaw optymalizatorów do trenowania sieci neuronowych. Podczas uczenia modelu różne optymalizatory oferują różne tryby dostrajania parametrów, często wpływając na zbieżność i wydajność modelu.
OBRAZ: Optymalizatory do trenowania sieci neuronowych.

7. TensorFlow - zastosowanie

TensorFlow został zaprojektowany przez naukowców i inżynierów Google, aby realizować projekty badawcze w dziedzinie maszynowego uczenia się *(machine learning)* oraz głębokich sieci neuronowych *(deep neural networks)*. System pozostaje mimo to dość ogólny, co pozwala na szeroką gamę innych zastosowań. TensorFlow umożliwia przede wszystkim optymalne uczenie się modeli wymagających dużej ilości danych (na przykład banków zdjęć).
TensorFlow służy jako podstawowa platforma i biblioteka do uczenia maszynowego. Interfejsy API TensorFlow wykorzystują Keras, aby umożliwić użytkownikom tworzenie własnych modeli uczenia maszynowego. Oprócz budowania i uczenia modelu, TensorFlow może również pomóc w załadowaniu danych w celu uczenia modelu i wdrożenia go za pomocą TensorFlow Serving. TensorFlow zapewnia stabilne API Pythona, a także API bez gwarancji kompatybilności wstecznej dla JavaScript, C++ i Javy.
OBRAZ: Podział sztucznej inteligencji *(artificial intelligence)* na uczenie maszynowe

(machine learning) i uczenie głębokie *(deep learning)*.
TensorFlow służy również do: rozpoznawania mowy, wyszukiwania głosowego, detekcji błędów, wykrywania języków, określania niebezpieczeństw w mediach społecznościowych, rozpoznawania obrazów, detekcji video.
OBRAZ: Rozpoznawanie mowy.
OBRAZ: Wyszukiwanie głosowe.
OBRAZ: Wykrywanie języków.
OBRAZ: Wykrywanie niebezpieczeństw w mediach społecznościowych.
OBRAZ: Rozpoznawanie obrazów.

8. CNN - splotowe sieci neuronowe

Splotowa (koewolucyjna) sieć neuronowa to pewien rodzaj układu sztucznych neuronów lub symulatorów neuronów, który ma działać w określony sposób. Sieci neuronowe to biologiczne grupy neuronów lub sztuczne grupy pseudo-neuronów, które są zaprogramowane do działania w taki sam sposób jak neurony biologiczne. Sztuczne sieci neuronowe starają się naśladować funkcje ludzkiego lub zwierzęcego mózgu.
W większości przypadków splotowa sieć neuronowa to po prostu sztuczna sieć neuronowa stworzona w celu symulacji pewnego rodzaju aktywności mózgu. Eksperci nazywają te modele „biologicznie inspirowanymi".
OBRAZ: Sieć neuronowa.
OBRAZ: Schemat działania splotowej sieci neuronowej.

9. CNN - splotowe sieci neuronowe

Ogólnie naukowcy, którzy wdrażają splotowe sieci neuronowe, odkryli niektóre ze specyficznych sposobów przetwarzania obrazów przez mózgi. Sztuczna inteligencja rozwinęła się w ostatnich czasach, a teraz naukowcy mogą sprawić, że technologie wykonują niektóre zadania, które kiedyś były zarezerwowane wyłączne dla widzenia biologicznego. Jednym z nich jest rozpoznawanie twarzy, gdzie zaawansowane algorytmy pozwalają aparatom i innym urządzeniom skutecznie wyświetlać obrazy i rozpoznawać indywidualną twarz.
Jednym z najbardziej rozpowszechnionych zastosowań splotowych sieci neuronowych jest symulacja widzenia ludzi lub zwierząt. Aplikacje te często koncentrują się na kombinacji danych wejściowych i wyjściowych, które pomagają technologii sztucznie robić to, co mózg robi naturalnie. Wiele złożonych metod, nazywanych warstwami, jest potrzebnych do osiągnięcia tego rodzaju symulacji. Są one często wyświetlane za pomocą modeli wizualnych, które pomagają czytelnikom zrozumieć, jak konfigurowana jest splotowa sieć neuronowa.
Koewolucyjne sieci neuronowe należy poddać rygorystycznym testom i ocenie pod kątem ich zalet, przy czym konkretne osiągnięcie wyników dowodzi, że technologie te mogą przynajmniej naśladować mózg człowieka lub zwierzęcia.
OBRAZ: Działanie splotowych sieci neuronowych w temacie rozpoznawania twarzy.

10. TensorFlow - CNN

TensorFlow poprzez integrację z biblioteką Keras posiada możliwość używania modeli sekwencyjnych. W tym celu używany jest interfejs API Keras Sequential. Pozwala to nam na zmniejszenie produkowanej ilości kodu jednocześnie przy dostępie do łatwej modyfikacji warstw.
OBRAZ: Logo Keras.
OBRAZ: Różne modele API - sekwencyjny interfejs API, funkcjonalny interfejs API, metody podklasy modelu obok siebie.

11. Część praktyczna

12. Dane

W naszym zadaniu przetwarzamy 3200 zdjęć MRI. Dotyczą one guzów mózgu, które są uważane za jedną z najbardziej agresywnych chorób wśród dzieci i dorosłych. Stanowią one od 85 do 90% wszystkich pierwotnych guzów ośrodkowego układu nerwowego (OUN). Każdego roku u ok. 11 700 osób diagnozuje się guza mózgu. Wskaźnik 5-letniego przeżycia dla osób z rakiem mózgu lub guzem OUN wynosi ok. 34% dla mężczyzn i 36% dla kobiet. Guzy mózgu mają różną klasyfikację - jako guz łagodny, guz złośliwy czy guz przysadki mózgowej. Aby poprawić długość życia pacjentów, należy wdrożyć odpowiednie leczenie, planowanie i dokładną diagnostykę.
OBRAZ: Zdjęcie MRI.
OBRAZ: Względne wskaźniki przeżycia (pierwotne guzy mózgu) według histologii w wieku 20+ lat (*glioblastoma* - glejak wielkopostaciowy (ostatnie stadium), *astrocytoma* - gwiaździak (astrocyty), *glioma* - glejak, *oligodendroglioma* - skąpodrzewiak (oligodendrocyty), *medulloblastoma* - rdzeniak zarodkowy (móżdżek), *ependymoma* - wyściółczak (tkanka wyścielająca światło komór mózgu i kanał środkowy rdzenia kręgowego)).

13. Dane

Najlepszą techniką wykrywania guzów mózgu jest obrazowanie metodą rezonansu magnetycznego (MRI). Ze skanów generowana jest ogromna ilość danych obrazu. Obrazy te są badane przez radiologa. Badanie manualne może być podatne na błędy ze względu na stopień złożoności guzów mózgu i ich właściwości.
Rezonans magnetyczny (MRI) wykorzystuje właściwości magnetyczne cząsteczek wody w organizmie człowieka, a ściślej protonów, w jądrach wodoru. Protony umieszczone w polu magnetycznym ulegają pewnemu uporządkowaniu. Poddane działaniu fal radiowych, o odpowiedniej częstotliwości, zmieniają swój stan i oddają energię wysyłając fale o tej samej częstotliwości (zjawisko rezonans). Fale są odbierane przez czujniki tomografu, a następnie komputerowo przetwarzane na przekrojowy obraz medyczny. Obróbka komputerowa pozwala również na stworzenie trójwymiarowych obrazów badanych narządów.
Rezonans magnetyczny w stosunku do badań radiologicznych z wykorzystaniem promieni rentgenowskich jest nieinwazyjny. Pole magnetyczne oraz fale radiowe wykorzystywane w tym badaniu są nieszkodliwe dla organizmu człowieka.
OBRAZ: Wykorzystywane przez nas zdjęcia określonych typów nowotworu mózgu: glejak (nowotwór wywodzący się z komórek glejowych stanowiących zrąb tkanki nerwowej i pełniącej wobec neuronów funkcje podporowe, odżywcze oraz naprawcze), osoba zdrowa, oponiak (nowotwór umiejscowiony w oponach mózgowo-rdzeniowych mózgowia lub rdzenia kręgowego), nowotwór przysadki (nowotwór związany często ze zmianą wydzielania określonych hormonów, np. prolaktyny czy kortykotropiny).

14. Nasze zadanie

Stosowanie technik automatycznej klasyfikacji przy użyciu uczenia maszynowego i sztucznej inteligencji konsekwentnie wykazuje wyższą dokładność niż klasyfikacja ręczna. Dlatego zaproponowanie systemu wykrywającego i klasyfikującego za pomocą algorytmów uczenia głębokiego z wykorzystaniem splotowych sieci neuronowych (CNN), sieci neuronowych (ANN) i Transfer Learning (TL) byłoby pomocne dla lekarzy na całym świecie.
Celem naszego zadania było utworzenie modelu do klasyfikacji pacjentów z

podejrzeniem rozwoju nowotworu mózgu na podstawie zdjęć MRI.
OBRAZ: Cel naszego działania.

15. Parametry używanych maszyn

Nasz kod sprawdzaliśmy na 3 maszynach - 2 z nich obejmowały typowy komputer PC a 1 maszynę wirtualną.
Pierwsze puszczenie kodu wykonaliśmy na procesorze CPU. Był to model AMD Ryzen serii 7 numer 2700 z 8 rdzeniami i 16 wątkami o podstawowym taktowaniu wynoszącym 3.2 GHz. Miał on 32 GB RAM typu GDDR4 o taktowaniu 3200 MHz.
W następnej kolejności wykorzystaliśmy kartę graficzną NVIDIA GTX 1660 z 1408 rdzeniami CUDA o taktowaniu 1830 MHz z 6 GB vRAM typu GDDR5. Procesor i podstawowy RAM były identyczne jak w pierwszym przypadku.
Jako ostatnią wynajęliśmy instancję maszyny wirtualnej od platformy Google Cloud Platform. Zbudowana była z procesora CPU Intel Xeon o 6 rdzeniach i 12 wątkach z taktowaniem podstawowym wynoszącym 2.0 GHz. Oprócz tego zawierała GPU NVIDIA A100-SXM4 z 6912 rdzeniami CUDA i 432 rdzeniami RT o taktowaniu 765 MHz z 40 GB pamięci vRAM typu HMB2. Dysponowała również 85 GB pamięci RAM.
OBRAZ: Używane maszyny wraz z parametrami.

16. Sposób działania

1. Import potrzebnych nam bibliotek i modułów.
2. Przygotowanie danych:
    1. Utworzenie list z nazwami przypadków w naszych danych.
    2. Utworzenie i wypełnienie dwóch list, w których znajdą się nasze obrazy, odpowiednio przeskalowane przy pomocy funkcji *resize* z modułu *cv2*.
    3. Odpowiednie podpisanie naszych obrazów - jaki jest to typ guza mózgu.
    4. Konwersja na typ *numpy.array*.
3. Podział naszych danych na zbiory testujący i uczący w proporcji 1:10 oraz zamiana etykiet na wartości liczbowe.
4. Pobranie modelu EfficientNetV2B0, z którego użyty zostanie zbiór danych ImageNet.
5. Odpowiednie ustawienie pobranego modelu - miejsce działania splotowych sieci neuronowych.
6. Kompilacja modelu.
7. Przed właściwym uczeniem modelu, odpowiednie ustawienie funkcji redukcji *„learning rate”*.
8. Trenowanie modelu.
9. Dokonanie predykcji, utworzenie wykresów oraz wypisanie ogólnej klasyfikacji.

17. Sposób działania

Przykładowe używane przez nas funkcje z paczki Keras z pakietu TensorFlow:
tf.keras.applications.EfficientNetV2B0 - zwraca model klasyfikacji obrazów, z możliwością załadowania wag ze wstępnie wytrenowanych modeli w ImageNet,
tf.keras.callbacks.TensorBoard - narzędzie do wizualizacji historii zmian w uczeniu modelu, zawiera on wykresy podsumowujące metryki, wizualizacje wykresu treningowego, histogramy aktywacji oraz profilowanie próbkowe, to funkcja wywoływana zwrotnie z *model.fit()*,
tf.keras.callbacks.ModelCheckpoint - wywołanie zwrotne razem z użyciem *model.fit()* w celu zapisania modelu wag z pewnego przedziału czasu,
tf.keras.callbacks.ReduceLROnPlateau - funkcja do zmniejszenia tempa uczenia się o współczynnik od 2 do 10 w momentach, gdy uczenie nie zwraca poprawy, też jest

wywołaniem zwrotnym przy użyciu *model.fit()*,
tf.keras.Model.fit() - trenuje ona model z określoną liczbą iteracji po zestawie danych, wykorzystuje wcześniej wspomniane funkcje, a przy ich braku ręcznego określenia tworzy je automatycznie, pozwala również określić sposób wyświetlania przebiegu uczenia modelu (argument 0 = tryb cichy, argument 1 = pasek postępu, argument 2 = jedna linia).
OBRAZ: Moduł Keras w paczce TensorFlow 2.0.

18. Prezentacja kodu

OBRAZ: Logo Tensorflow.
OBRAZ: Logo Python.

19. Wyniki

Nasze wyniki obejmują 2 wykresy obrazujące skuteczność naszego uczenia.
Wykres po lewej dotyczy zależności liczby trenowań modelu od dokładności wyników. Inaczej mówiąc - jest to porównanie wyniku otrzymanego za pomocą modelu z wartością rzeczywistą.
Wraz ze wzrostem liczby trenowań modelu wzrasta poziom dokładności dla procesu uczenia (kolorem zielonym). Możemy również zobaczyć linię koloru czerwonego dotyczącą dokładności walidacji czyli wyników na zbiorze testowym. Otrzymywane wartości są niższe ze względu na potrzebę weryfikacji danych, aby nie doprowadzić do przetrenowania modelu względem konkretnych danych - musi być on uniwersalny względem ich wszystkich. W obu przypadkach oczekiwane wyniki widzimy już przy 5 powtórzeniu - dokładność uczenia wynosi niemalże 100%, a dokładność walidacji osiąga próg 96%.
Drugi wykres dotyczy liczby trenowań modelu w porównaniu ze stratami w uczeniu i walidacji. Utrata podczas szkolenia lub walidacji to metryka używana do oceny dopasowania modelu uczenia głębokiego do danych uczących lub testowych - ocenia ona błąd modelu. Stratę uczącą oblicza się, biorąc sumę błędów dla każdego przykładu w analizowanym zbiorze.
Wraz ze zwiększeniem liczby trenowań modelu znacząco spada liczba pomyłek. Jest to spowodowane większym dostosowaniem modelu do danych, tak aby wykrywał dla nas istotne elementy, świadczące o obecności konkretnego rodzaju nowotworu bądź jego braku. Identyczny trend dotyczy również procesu walidacji. Tutaj również należy uważać na przetrenowanie modelu, aby mógł być wykorzystany także do innych danych.
OBRAZ: Wykresy wynikowe.

20. Wyniki

Oprócz wyników w formie wykresów uzyskaliśmy mapę cieplną dokładności modelu. Jest ona wykorzystywana do przedstawienia wizualizacji pracy algorytmu (najczęściej przy uczeniu nadzorowanym). Mapa cieplna przedstawia dokładność klasyfikacji obrazów MRI w odniesieniu do mocy modelu. Moc testu to prawdopodobieństwo uniknięcia błędu drugiego rodzaju - przyjęcia hipotezy zerowej, gdy w rzeczywistości jest ona fałszywa. W naszym przypadku oznacza ona brak rozpoznania choroby u pacjenta - pomimo występowania, pacjent nie zostaje rozpoznany względem danego schorzenia, co wydaje się znacznie gorsze niż błąd pierwszego rodzaju (wykrycie schorzenia przy jego braku).
Największą szansę wykrycia ma nowotwór przysadki, dla którego moc modelu wynosi aż 96, dla oponiaka moc modelu wynosi 93. Może zostać on najczęściej błędnie zinterpretowany jako glejak, dla którego to nasz model ma najmniejszą moc

wynoszącą 77. Ponadto dla osoby zdrowej moc modelu wynosi 93.
OBRAZ: Mapa cieplna dokładności modelu.

21. Wnioski

Za pomocą paczki TensorFlow udało się przygotować model rozpoznający odpowiedni rodzaj nowotworu mózgu. Na skutek wrzucenia dużej liczby zdjęć MRI nastąpił proces *machine learningu*. Należy zauważyć, że zastosowaliśmy tutaj klasyfikację nadzorowaną - na samym początku wskazywaliśmy odpowiednie wyniki dla poszczególnych zdjęć.
Wraz ze wzrostem liczby trenowań naszego modelu, jego dokładność dla zbiorów uczącego oraz testowego wzrastała, natomiast liczba pomyłek spadała. Świadczy to o prawidłowym postępie procesu uczenia.
Mapa cieplna modelu również potwierdziła jego skuteczność, której wysoki wskaźnik jest niezbędny w badaniach medycznych. Istotna jest również moc testu, której wysoka wartość umożliwia poprawne stwierdzenie schorzenia i podjęcie właściwego leczenia.
Optymalny model uzyskujemy już przy 5. powtórzeniu. Większa liczba powtórzeń trenowania naszego modelu nie przekładała się na lepsze wyniki, a wręcz powodowała stopniowe jego przetrenowanie.
OBRAZ: Wysoka dokładność.
OBRAZ: Wysoka skuteczność.

22. Inne możliwości

Odpowiednikiem użytej biblioteki TensorFlow w środowisku Python jest PyTorch. Pozwala on również na uruchamianie obliczeń na GPU z wykorzystaniem architektury CUDA, czyli do akcelerowanej przez GPU biblioteki cuDNN *(NVIDIA CUDA Deep Neural Network)*.
Inną możliwością było zastosowanie Scikit-learn. Jest on również wykorzystywany do uczenia maszynowego w środowisku Python, jednak w odróżnieniu od TensorFlow, obliczenia prowadzi tylko na jednostce głównej CPU komputera.
OBRAZ: Logo PyTorch.
OBRAZ: Logo Scikit-learn.

23. Podsumowanie

Moduł Keras zaimplementowany w pakiecie TensorFlow w języku Python służy do uczenia maszynowego, dzięki czemu jesteśmy w stanie stworzyć swoją sieć neuronową.
Splotowa sieć neuronowa (CNN) wykorzystywana jest z powodzeniem m.in. w rozpoznawaniu obrazów, co ma zastosowanie w szerokiej gamie dziedzin badawczych.
Celem uzyskania najlepszego modelu należy wykonać większą liczbę powtórzeń przyrównań, uważając aby nie przetrenować otrzymanego modelu.
Wykorzystanie karty graficznej wielokrotnie przyspiesza proces otrzymania właściwych modeli.
Odpowiednio wysoki stopień predykcji jest niezwykle ważny w medycynie, w której to niedopuszczalne jest pominięcie właściwej klasyfikacji osoby chorej (błąd II rodzaju).
OBRAZ: Wniosek 1.
OBRAZ: Wniosek 2.
OBRAZ: Wniosek 3.
OBRAZ: Wniosek 4.
OBRAZ: Wniosek 5.

24. Slajd zakończeniowy