"Klasyfikacja guzów mózgu w obrazach 3D wykonanych z wykorzystaniem techniki rezonansu magnetycznego (MRI)".

W pierwszej części projektu skupiam się na opracowaniu (przygotowaniu) zestawu danych, na których będę pracować podczas analizy.

Zestaw danych, na których pracuję pochodzi ze strony:
https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri?resource=download
A wgląd do zestawów: treningowego i testowego jest możliwy pod linkiem:
https://github.com/sartajbhuvaji/brain-tumor-classification-dataset

Artykuły, do których się odwołuję:

1. "Classification of brain tumours in MR images using deep spatiospatial models" -> https://www.nature.com/articles/s41598-022-05572-6#Sec5
2. "3D MRI brain tumor segmentation using autoencoder regularization" -> https://arxiv.org/pdf/1810.11654v3.pdf

Zestaw jest podzielony na dwie części (jak wyżej), można dodatkowo wyróżnić jeszcze zbiór walidacyjny, ale względem pierwszego artykułu, będę korzystała z tego podziału jaki jest - treningu i testu.

Przypadki badanych guzów w obrazie MRI są podzielone na 4 klasy: glejaki, oponiaki, gruczolaki, oraz przypadki bez nowotworu.

Problemem, na którym chciałabym się skupić w projekcie jest dopasowanie fotografii (klasyfikacja) danych guzów mózgu (lub braku) do odpowiedniej z czterech klas.

ETAP I

Przywołuję sobie dane z dysku, wydzielam sobie podgrupy - treningowa i testowa. Przyjmuję założenie, że klasy są odpowiednikami nazw folderów. Przekształcam nazwy klas na liczby.
Następnie tworzę listę, która bierze pod uwagę zdjęcia ze zbioru treningowego. Lista zawiera krotki (ścieżka zdjęcia, klasa zdjęcia).
Wyznaczam wielkość zbioru. Wiem, że jest mniejszy niż 4 tysiące obrazów, dlatego ustalam, że nie może on przekroczyć tej wartości. Potem wyznaczam obrazy ze ścieżki.
Z racji tego, że Dataset stanowią obrazy to wykonuję ujednoliczenie do rozmiaru 224x224, w skali szarości, w zakresie 0-1.
Hiperparametr w tym przypadku ustalam na 16 (może być też 32, w pierwszej publikacji badano to na 2 sposoby).
Sprawdzam czy działa i wczytuje przypadki uwzględnione z pierwszej komórki.

Źródła kodów:
https://www.youtube.com/watch?v=oWq6aVv5mC8
https://pytorch.org/tutorials/beginner/basics/data_tutorial.html
https://stackoverflow.com/questions/33533148/how-do-i-type-hint-a-method-with-the-type-of-the-enclosing-class

Potem robię przypasowanie obrazów do słownika. Liczby odpowiadają tym z dopasowania wyniku z pierwszej komórki.

ETAP II

Implementacja sieci konwolucyjnej. Korzystam z powielonego do warstw kodu ze strony: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html oraz z naszych zajęć nr 6, bo również mieliśmy zadanie związane z CIFAR10 i CNN.
Porównam sobie 2 sposoby, ponieważ nie byłam pewna, na podstawie wyników w artykule oraz znalezionych w internecie czy wyszło mi to dobrze.
Drugi sposób - korzystam z:
https://www.kaggle.com/code/priyanshzalavadiya/cnn-with-100-training-accuracy
W przypadku pierwszym ustalam sobie 9 warstw (w przykładach z publikacji spotkałam się także z 4 oraz 7). Każda z warstw składa się z warstwy konwolucyjnej, normalizującej i ReLU, tylko ostatnia uśrednia (Average Pooling).
Wywołuję krok forward. Korzystam z funkcji sigmoid, w zależności od publikacji wykonywali tym lub softmax, dlatego w drugim sposobie mam softmax, żeby to porównać.

ETAP III

Trenuję sieć w sposób nadzorowany przypadek → klasa. W przypadku danych medycznych wymaga to większej uwagi, ze względu na podobieństwo struktur. Dobieram parametry: współczynnik uczenia na poziomie 0.0003 (w publikacji także 0.0001), rozmiar batcha 32 (może być też 40). Epoki - w publikacji jest to 50, niestety moc przerobowa mojego komputera dochodziła do max 27 epok. Wzięłam pod uwagę 5-10 epok w zależności od sposobu implementacji sieci. Patrząc po wynikach było to wystarczające, dzięki temu nie było też problemu z przetrenowaniem sieci. Wybrałam optymalizator Adam oraz urządzenie. Według publikacji "cuda:0", natomiast sieć albo nie chciała się liczyć albo była niestabilna. Zamieniłam na “cpu”. Szybciej się nie liczyła, ale przynajmniej dała wynik.

Trenuję model, wyznaczam funkcję kosztu, aktualizuję optymalizator i zeruję gradient.

ETAP IV

Analizuję wyniki, wybrałam macierz pomyłek. W drugiej wersji sieci sprawdzam jedynie dopasowanie. Macierz pomyłek klasyfikuje mi dane przypadki w zależności od rodzaju guza i jego poprawnego (lub fałszywego) dopasowania. Poza mapami sprawdzam jakie były pomyłki (Pre - przewidywana, Trg - jaka powinna być). Więcej na ten temat w prezentacji na zajęciach.

Podsumowanie:
Modele nie są najlepsze, gdyby je dalej trenować (lub chociażby porównując wyniki do publikacji) to doszłoby do całkowitego przetrenowania, a dopasowanie by się nie zgadzało. Testując już chociażby na różnej liczbie epok (nie biorąc pod uwagę hiperparametru), daje różne dopasowanie.