Transfer learning w klasyfikacji obrazów

Daniel Ząbek

Politechnika Wrocławska 259778@student.pwr.edu.pl

Streszczenie Transfer learning to popularna technika wykorzystywana do tworzenia różnych modeli, które bazują na popularnych i dobrze opracowanych i przebadanych modelach. Takie podejście do problemu zaoszczędza dużo czasu przy budowie architektury, ale także daje nam możliwość używania wag modelu, które są zazwyczaj wynikiem trenowania modelu na bazie obrazów z ImageNet, co daje nam możliwość wykorzystania cech wyuczonych na jednym problemie i wykorzystanie ich w nowym, podobnym problemie. Ta praca będzie opierać się na użyciu modelu EfficientNet do klasyfikacji różnych typów guzów mózgu.

Keywords: Transfer Learning · Brain Tumors · Regularization · Data Augumentation

1 Studia literaturowe

Transfer learning jest bardzo popularną metodą szczególnie w analizie obrazów medycznych, dzięki zmniejszeniu ilości danych potrzebnych do wytrenowania CNN [1]. W przypadku baz danych typu guzy mózgu, gdzie ilość naszych danych jest ograniczona rozbudowane modele często doprowadzają do nadmiernego dopasowania (overfitting), w którym nasz model zapamiętuje naszą bazę danych zbyt dokładnie razem z wszelkimi szumami, zamiast nauczyć się ich najważniejszych cech w celu klasyfikacji. Popularną techniką jest augumentacja danych, która polega na realistycznych transformacjach naszych obrazów w celu zwiększenia generalizacji modelu. Na przykład, w celu zwiększenia dokładności segmentacji i klasyfikacji guzów w modelach zastosowano dodanie szumu i wyostrzenie obrazów [2].

Dropout to jedna z popularnych technik regularyzacji dla głębokich sieci neuronowych, została wprowadzona przez Geoffreya Hintona w 2012 roku [3] i rozwinięta w późniejszej pracy Nitisha Srivastavy et al. [4]. O jej sukcesie świadczy fakt, że dodanie Dropout doprowadziło do zwiększenia dokładności o 1-2% nawet w zaawansowanych sieciach neuronowych [5].

Następna technika często wykorzystywana to Batch Normalization.Jeśli chodzi o głębokie sieci neuronowe z wieloma warstwami, każda z własnymi aktywacjami, Batch normalization jest używany do normalizacji aktywacji, aby poprawić uczenie kolejnych parametrów [6].

2 Przygotowanie do eksperymentu

2.1 Eksploracja danych

Dane składają się z 7022 obrazów guzów mózgu: Folder to testowania prezentuję się następująco:

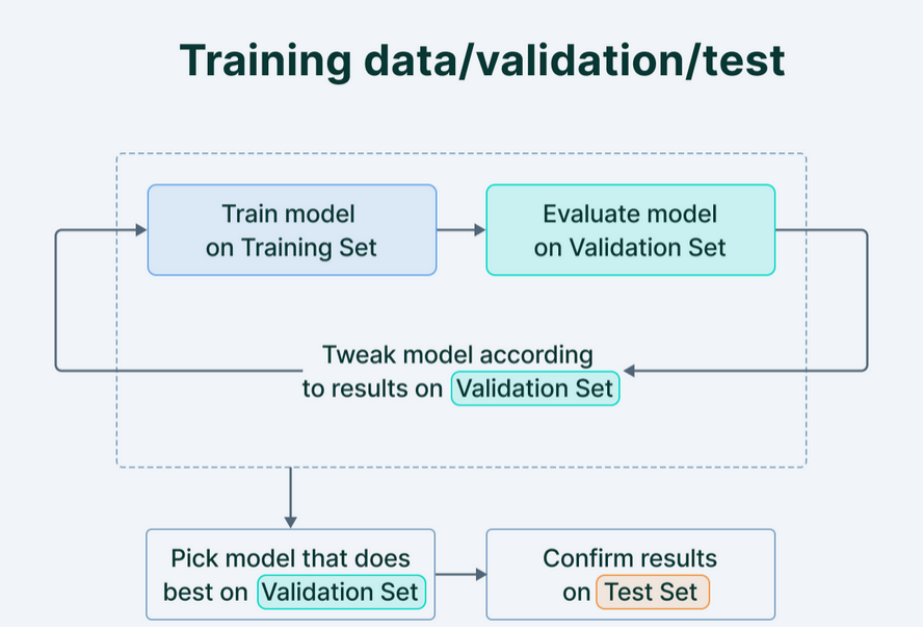
- Meningioma: 306 obrazów
- Glioma: 300 obrazów
- Pituitary: 300 obrazów
- Notumor: 405 obrazów

Folder do trenowania modelu:

- Meningioma: 1339 obrazów
- Glioma: 1321 obrazów
- Pituitary: 1457 obrazów
- Notumor: 1595 obrazów

2.2 Podział danych na zbiór trenowania, walidacji i testowania

W zbiorze danych oryginalny folder testowy zostały podzielony na pół, w wyniku czego powstał zbiór walidacyjny zawierający 655 obrazów, a pozostałe 656 obrazów tworzyło zestaw testowy dla guzów mózgu. Dane przed podziałem zostały wymieszane, aby zapobiec błędu obciążenia (bias). Rysunek 1 poniżej dobrze przedstawia metodologie uczenia maszynowego, używając tych trzech zbiorów.



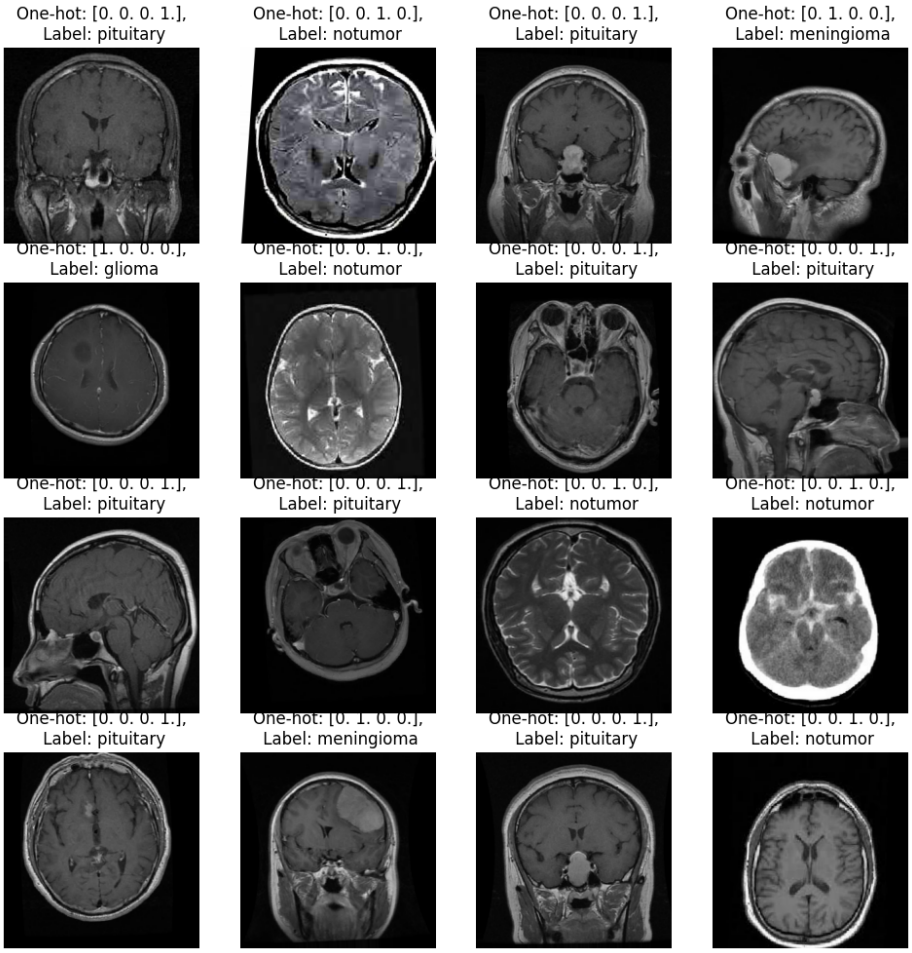
Rysunek 1. Metodologia uczenia modelu [8].

2.3 Augumentacja Danych

Augumentacja danych wprowadza zmienność, dzięki której model jest bardziej odporny i zdolny do generalizacji na nieoczekiwane przykłady. Pomaga to zapobiegać nadmiernemu dopasowaniu, czyli zapobieganiu uczenia się na pamięć danych z grupy do trenowania, co doprowadza do złych wyników przy ewaluacji modelu na danych z grupy do testowania. Jeśli nasz model zacznie zapamiętywać zbiór danych treningowych będzie także zapamiętywać niepożądane elementy, które nie mają znaczenia przy klasyfikacji różnych klas, lub zacznie także uczyć się szumu, oraz różnych artefaktów w obrazach zamiast ich najważniejszych elementów, które świadczą o danej chorobie. Użyte transformacje na obrazach to:

- rotacja o 5 stopni
- przesunięcie poziome o max. 5% szerokości
- przesunięcie pionowe o max. 5% wysokości
- odwrócenie poziome

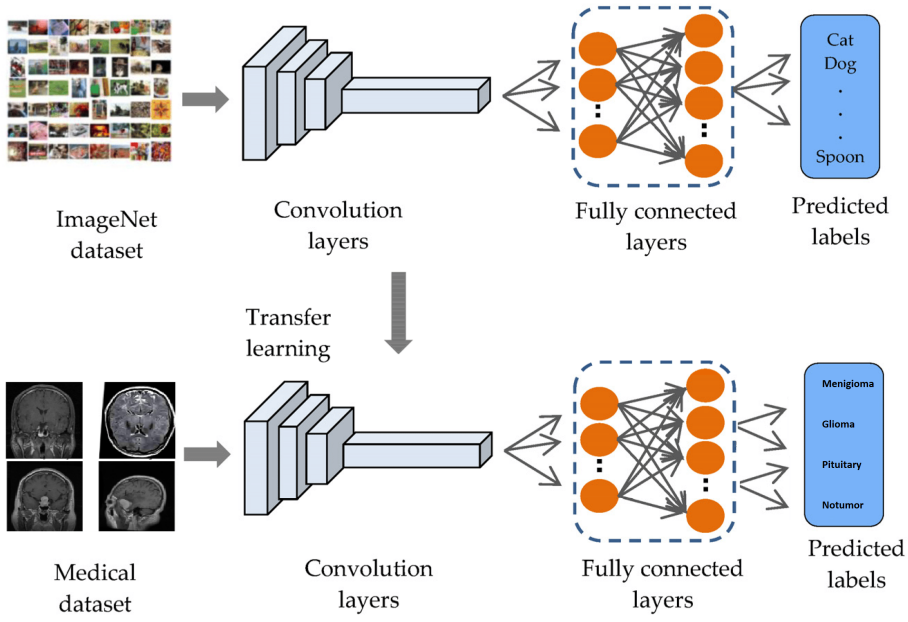
Jeden batch danych treningowych składający się z 16 obrazów został przedstawiony na rysunku 2.



Rysunek 2. Jeden batch danych treningowych po augumentacji

2.4 Transfer learning

Transfer learning polega na wykorzystaniu wytrenowanego modelu na innych danych na przykład obrazach z ImageNet i przetransferowanie nauczonej wiedzy na ich podstawie do naszego specyficznego zadania, w tym przypadku klasyfikacji guzów mózgu, zobrazowanie tego procesu przedstawiono na rysunku 3. Jest to bardzo popularna technika szczególnie w medycynie z oczywistych powodów, takich jak limitowana ilość dobrze opisanych danych [1]. Możliwe jest również wykorzystanie modelu i trenowanie jego wag na podstawie tylko naszych danych. Zazwyczaj usuwamy top-layers z modelu i dodajemy nasze w celu dalszej optymalizacji modelu do naszego danego zadania. Praca ta skupi się na optymalizacji wyników modelu dla zadania klasyfikacji skanów MRI mózgu.



Rysunek 3. Zobrazowanie głównej idei transfer learning

2.5 Plan eksperymentu

2.6 Główny cel pracy

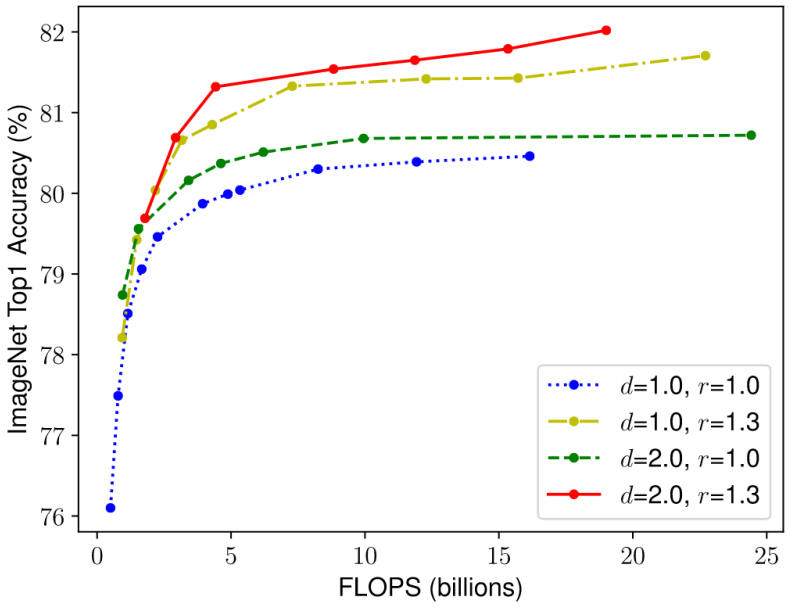
Eksperyment będzie polegał na wykorzystaniu relatywnie nowej architektury EfficientNet [7]. Zostanie wybrana wersja, która będzie odpowiednia dla danego zbioru danych, oraz model zostanie dalej poddany optymalizacji (fine-tuning) poprzez zastosowanie technik regularyzacji, aby zwiększyć generalizacje naszego modelu dla nowych danych. Model będzie ewaluowany na zbiorze walidacyjnym podczas treningu, oraz ostatecznie na zbiorze testowym, czyli obrazach nigdy wcześniej nie widzianych przez nasz model. Głównym pytaniem pracy jest, czy jesteśmy w stanie na podstawie modelu Efficientnet, który posiada znacznie mniej parametrów stworzyć model osiągający wyniki zbliżone do notebooka, który wygrał złoty medal w konkursie organizowanym przez Kaggle posiadając dokładność 99.24% i stratę 0.0492, z użyciem 21 124 268 parametrów (80.85 MB) [9].

2.7 EfficientNet

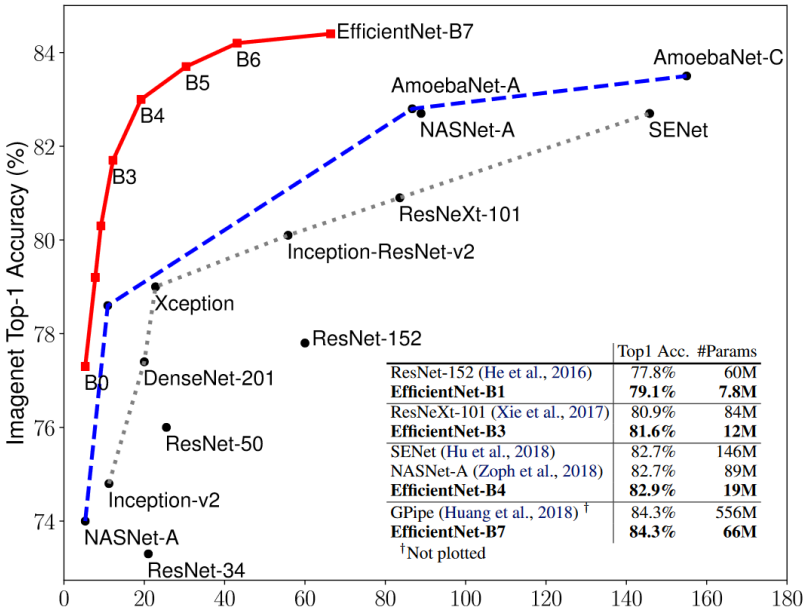
Sekcja ta opisuje dokładnie model EfficientNet na podstawie "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" [7]. EfficientNet stara się uwzględnić równowagę między:

- Głębokością (depth) - ilość warst w sieci.
- Szerokością (width) - ilość kanałów w warstwach.
- Rozdzielczość (resolution) - wielkość naszego wejściowego obrazu.

Osiąga tę równowagę poprzez zastosowanie metody nazwaną "Compound Scaling", która równomiernie skaluje rozdzielczość, głębokość i szerokość sieci neuronowej, co różni się od zwykłych podejść. W poprzednich pracach często skaluje się tylko jedn z trzech wymiarów - głębokość, szerokość i rozmiar obrazu. Skalowanie tych wymiarów arbitralnie wymagało wielu prac manualnych i często prowadziło do gorszej dokładności, ukazuje to rysunek 4. Artykuł na temat EfficientNetB przedstawia metodę skalowania sieci neuronowych, która może osiągnąć lepszą dokładność i efektywność dzięki zastosowanej metodzie przy jednoczesnym użyciu mniejszej liczby parametrów, niż wiele innych modelów jak ukazuje rysunek 5.



Rysunek 4. Dokładność przy skalowanie pojedynczych parametrów kontra wielu jednocześnie [7].



Rysunek 5. Porównanie dokładności i ilości parametrów między modelami [7].

Wcześniej wspomniane podejście regulacji wszystkich trzech parametrów zapewnia zrównoważoną zmianę trzech parametrów, aby żaden z nich nie dominował, co prowadzi do lepszej wydajności. Wykorzystano współczynnik ϕ do równomiernego skalowania głębokości sieci, jej szerokości i rozdzielczości w przedstawiony sposób:

$$\begin{aligned} \text{depth: } & d = \alpha^\phi \\ \text{width: } & w = \beta^\phi \\ \text{resolution: } & r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \quad \beta \geq 1, \quad \gamma \geq 1 \end{aligned}$$

Jak można wywnioskować skalujemy proporcjonalnie wszystkie trzy parametry. Równanie $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ jest ograniczeniem, które ma na celu zagwarantowania nam zwiększenia FLOPS o czynnik 2^ϕ, co daje łatwą kalkulację kosztów obliczeniowych modelu.

3 Tworzenie i optymalizacja modelu

3.1 Batch Normalization

Pokazano w oryginalnej pracy wprowadzającej batch normalization autorstwa Sergeya Ioffe'a i Christiana Szegedy'ego [10], główna idea może być podsumowana równaniami przedstawionymi na rysunku 6.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

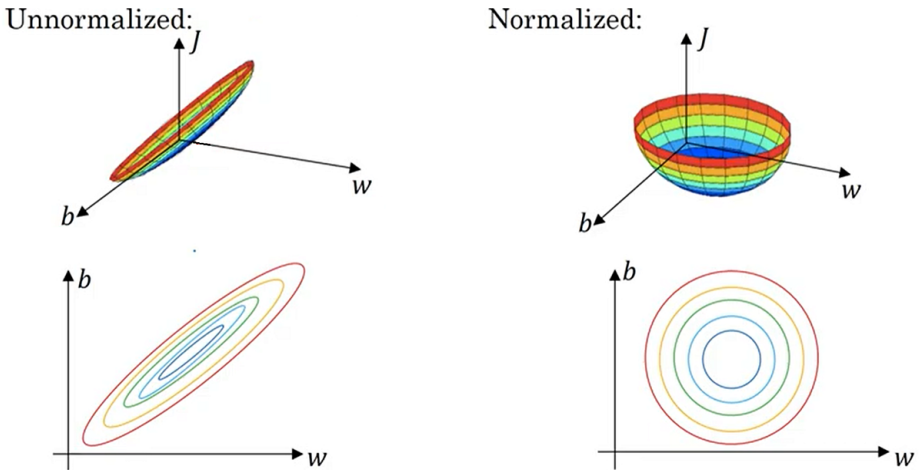
$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Rysunek 6. Równania przedstawiające Batch Normalization [10].

Analizując równania możemy dojść do wniosku, że W każdej iteracji treningowej najpierw normalizujemy wejścia, odejmując ich średnią i dzieląc przez ich odchylenie standardowe, gdzie oba są oszacowane na podstawie statystyk bieżącego mini-batch. Następnie stosujemy współczynniki skali i przesunięcia, aby odzyskać utracone stopnie swobody. Po zastosowaniu standaryzacji, otrzymany mini-batch ma średnią równą zero i wariancję jednostkową (unit variance) [11]. W oryginalnym artykule autorzy stwierdzili, że efektywność tej metody polega na zredukowaniu "Internal covariate shift", lecz późniejsze artykuły zaprzeczają tym założeniu. Na dzień dzisiejszy przyjęte jest, że daje to efekt zmiany kształtu naszego problemu uczenia, co przedstawiono na rysunku 7. Ten zabieg pozwala na łatwiejsze uczenie modelu i mniejsze znaczenie inicjalizacji jego wag. 'Indeed, we identify the key impact that BatchNorm has on the training process: it reparametrizes the underlying optimization problem to make its landscape significantly more smooth.' [12]



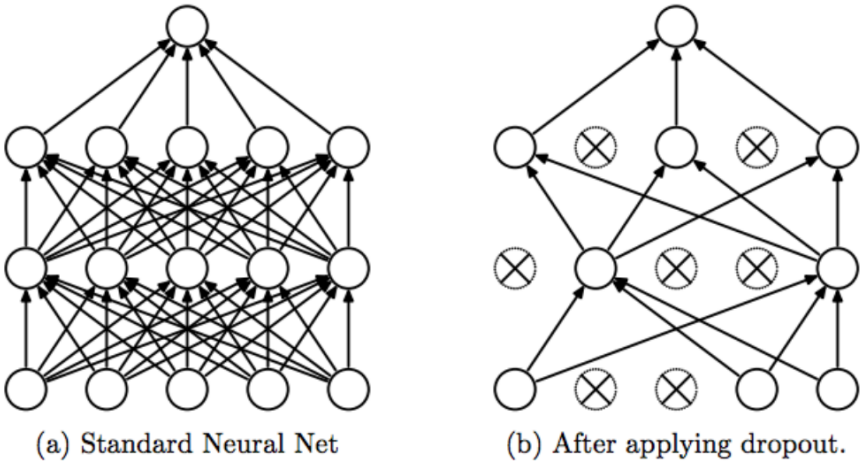
Rysunek 7. Uproszczone przedstawienie wygładzenia przestrzeni problemu.

3.2 Dropout Layer

Dropout jest techniką regularizacji stosowaną w sieciach neuronowych, która polega na losowym wyłączaniu (zerowaniu) pewnej liczby neuronów w warstwie podczas treningu, jak pokazuje rysunek 8. Prawdopodobieństwo wyłączenia neuronu jest określone przez hiperparametr p.

Sens tej techniki wynika z tego, że wyłączając losowo niektóre neurony w trakcie treningu, wymuszamy sieć nauczyć się bardziej odpornych i niezależnych cech, co może poprawić ogólną zdolność generalizacji modelu. Innymi słowy, dropout zapobiega zbytniemu dostosowaniu się sieci do danych treningowych

poprzez eliminację nadmiernych zależności pomiędzy neuronami, co może prowadzić do overfittingu dzięki większemu rozproszeniu wag. [13]



Rysunek 8. Wizualizacja zastosowania dropout [13].

3.3 Regularyzacja L2

Zamiast bezpośrednio manipulować liczbą parametrów, regularyzacja poprzez "weight decay"działa poprzez ograniczanie wartości, jakie mogą przyjąć parametry. Nazywana zazwyczaj regularyzacją $L2$. Technika ta jest motywowana podstawową intuicją, że spośród wszystkich funkcji f, funkcja $f = 0$ (przypisująca wartość do wszystkich wejść) jest w pewnym sensie najprostsza i możemy mierzyć złożoność funkcji przez odległość jej parametrów od zera. Poprzez dodanie "penalty term", opartego na wielkości współczynników do funkcji straty, model jest zniechęcany do przypisywania zbyt dużej wagi do indywidualnej cechy. W innych słowach duże wagi modelu są proporcjonalnie karane przez "penalty term", co prowadzi do mniejszych wag i uproszczenia modelu, a za tym mniejszej szansy na overfitting [14].

3.4 Ostateczny model

Ostateczny model pokazany na rysunku 9 z użyciem opisanych technik zawiera 11 184 179 parametrów (42.66 MB)

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb3 (Functional  (None, 1536)              10783535
 )

 batch_normalization (Batch  (None, 1536)              6144
 Normalization)

 dense (Dense)               (None, 256)               393472

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 4)                 1028

=================================================================
Total params: 11184179 (42.66 MB)
Trainable params: 11093804 (42.32 MB)
Non-trainable params: 90375 (353.03 KB)
_________________________________________________________________
```

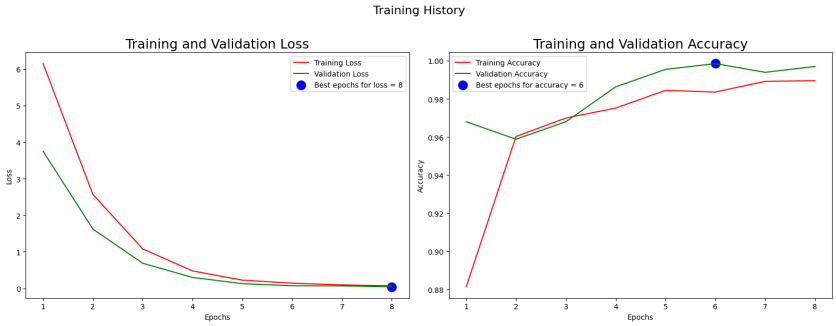
Rysunek 9. Ostateczny model.

4 Wyniki badań

Model osiągnął **99.02%** dokładności na danych testowych, jest to niemal taka sama dokładność jak zwycięzca złotego medalu w konkursie Kaggle przy jednoczesnym użyciu 9 940 089 mniej parametrów, spadek o **47.06%** i **38.19** MB mniej pamięci, spadek o **47.24%**.

4.1 Wykresy dokładności i strat

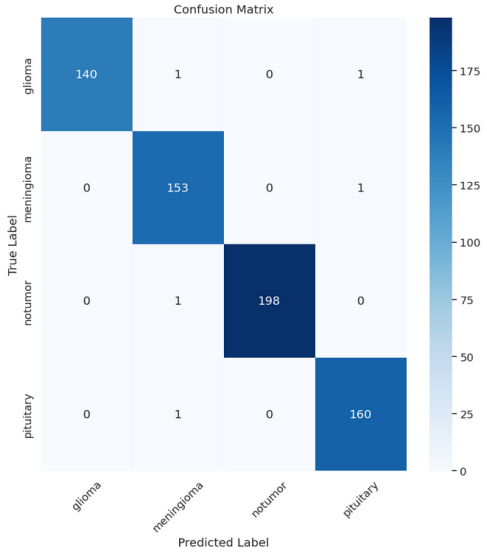
Na podstawie wykresu 10 na rysunku możemy zobaczyć, że model odpowiednio uczył się podczas trenowania i nie było problemu z overfitting, wyniki dla danych walidacyjnych były zbliżone.



Rysunek 10. Wykresy dokładności i strat podczas trenowania.

4.2 Confusion Matrix i report klasyfikacji

Macierz błędu na rysunku pozwala nam wizualnie zobaczyć w łatwy sposób z jakimi klasami model miał problemy i w jaki sposób je pomylił (True Positive, False Positive, True Negative, False Negative).



Rysunek 11. Macierz błędu.

Na podstawie True Positive, False Positive, True Negative, False Negative wyliczane są:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN}$$

Na ich podstawie generowany jest ostateczny raport klasyfikacji przedstawiony na rysunku 12.

	precision	recall	f1-score	support
glioma	1.00	0.99	0.99	142
meningioma	0.98	0.99	0.99	154
notumor	1.00	0.99	1.00	199
pituitary	0.99	0.99	0.99	161
accuracy			0.99	656
macro avg	0.99	0.99	0.99	656
weighted avg	0.99	0.99	0.99	656

Rysunek 12. Raport klasyfikacji.

5 Wnioski

Zastosowanie nowych modeli w celu klasyfikacji, przy odpowiedniej optymalizacji jest w stanie dać nam bardzo wysoką dokładność, przy zastosowaniu niewielkiej liczby parametrów. Zrozumienie obecnych technik regularyzacji i zapobiegania nadmiernego dopasowania jest niezbędne do tworzenia dobrych klasyfikatorów. Do klasyfikacji w tej pracy EfficientNetB3, był najbardziej skuteczny dając **99.02%** dokładności i najmniej podatny na overfitting, jednocześnie używając tylko **42.66 MB** pamięci.

Literatura

1. Salehi AW, Khan S, Gupta G, Alabduallah BI, Almjally A, Alsolai H, Siddiqui T, Mellit A. A Study of CNN and Transfer Learning in Medical Imaging: Advantages,

Challenges, Future Scope. Sustainability. 2023; 15(7):5930. https://doi.org/10.3390/su15075930
2. Khan, Amjad Rehman, et al. "Brain tumor segmentation using K-means clustering and deep learning with synthetic data augmentation for classification."Microscopy Research and Technique 84.7 (2021): 1389-1399.
3. Hinton, Geoffrey E., et al. "Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors." ArXiv:1207.0580 [Cs], 3 July 2012. https://arxiv.org/abs/1207.0580
4. Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Nitish Srivastava and Geoffrey Hinton and Alex Krizhevsky and Ilya Sutskever and Ruslan Salakhutdinov, 2014. http://jmlr.org/papers/v15/srivastava14a.html
5. Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. Concepts, Tools, and Techniques to Build Intelligent Systems 2nd Edition, p. 357, 2019.
6. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." https://arxiv.org/abs/1905.05928
7. Tan, Mingxing, and Quoc V. Le. "efficientnet: Rethinking model scaling for convolutional neural networks." arxiv:1905.11946 [cs, stat], 11 sept. 2020 https://arxiv.org/abs/1905.11946
8. Baheti, Pragati "Train, validation, and test set: How to split your machine learning datahttps://www.v7labs.com/blog/train-validation-test-set
9. Gold medal winning kaggle notebook https://www.kaggle.com/code/yousefmohamed20/brain-tumor-mri-accuracy-99
10. Ioffe Sergey, and Christian Szegedy. "batch normalization: Accelerating deep network training by reducing internal covariate shift"https://arxiv.org/abs/1502.03167
11. Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J. Book Title Dive into Deep Learning p. 294, publisher Cambridge University Press, 2023
12. How Does Batch Normalization Help Optimization? Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, Aleksander Madryhttps://arxiv.org/abs/1805.11604
13. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting https://jmlr.org/papers/v15/srivastava14a.html
14. Zhang, Aston and Lipton, Zachary C. and Li, Mu and Smola, Alexander J. Book Title Dive into Deep Learning p. 119-121, publisher Cambridge University Press, 2023