Szymon Walkusz 215 IC_b2

Cyfrowe Przetwarzanie Obrazów

Temat projektu: *Wykorzystanie sztucznej inteligencji do rozpoznawania obrazów*

Wstęp

Projekt prezentuje sposoby klasyfikowania obrazów na podstawie Konwolucyjnych sieci neuronowych (CNN), przy użyciu augmentacji danych, klasyfikacji binarnej, na podstawie wytrenowanej wcześniej sieci z użyciem zamrażania warstw i architektury sieci VGG16.

Opisy stosowanych technologii, architektury oraz przebieg zadań są opisane na bieżąco w niniejszym sprawozdaniu.

Projekt podzielono na klika metod klasyfikacji:

1. Model oparty tylko na dostępnych danych - bez mechanizmu regularyzacji
2. Model oparty o augmentację danych
3. Model oparty o dostrajanie uprzednio trenowanej sieci

Pliki z rozszerzeniem *.py* zawierają skrypty podzielone na sekcje, wraz z opisami.

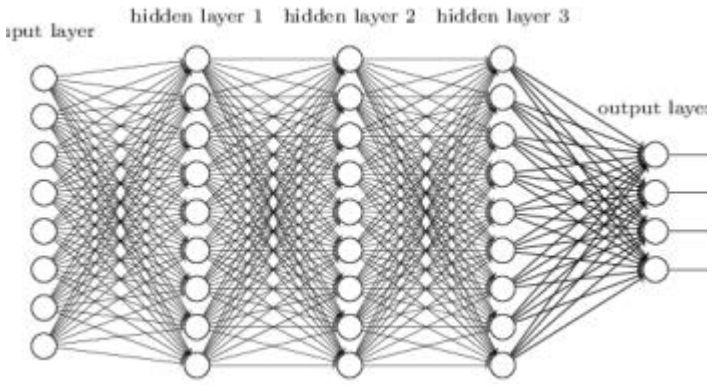
Plik *szymon_walkusz.py* zawiera kod odnoszący się do pierwszych dwóch technik, natomiast *VGG16.py* odnosi się do ostatniej metody.

W folderze również zapisano wytrenowane modele.

Wprowadzenie do Deep Learningu

Użycie sztucznych sieci neuronowych w ostatnich latach staje się coraz bardziej popularne, szczególnie podczas detekcji detekcji obiektów na obrazach cyfrowych. Zastosowanie to występuje niemalże w każdej dziedzinie np. w medycynie, przemyśle motoryzacyjnym, systemach bezpieczeństwa, grach komputerowych.

Modele detekcji implementują wyspecjalizowane algorytmy, często połączone w różne konfiguracje z innymi modelami. Sieci głębokie charakteryzują się liczną liczbą warstw sieci, gdzie w każdej warstwie może występować wiele neuronów. Często można się spotkać z określeniem, że “sieć posiada wiele warstw ukrytych”. *Rys. 1* ilustruje przykład CNN.



Rys. 1. Przykład CNN

Popularnymi rozwiązaniami przy tworzeniu sieci neuronowych są m.in:

- CNN (Convolutional Neural Networks) – Neuronowe Sieci Konwolucyjne - stosowane w problemach klasyfikacji obrazów składają się z trzech elementów:
  - serii warstw konwolucyjnych i łączących (pooling)
  - gęsto połączony klasyfikator na końcu
- HOG (Histogram of Oriented Gradients) - powszechnie używany do ekstrakcji cech z obrazu takich jak kształt
- VGG16 – architektura ta składa się z 16 warstw, które mają trenowane parametry (13 warstw konwolucyjnych i 3 w pełni połączone warstwy)
- VGG19 – jest rozwinięciem architektury VGG16

Warstwy Sieci Neuronowych:

- Warstwa Wejściowa: Otrzymuje surowe dane wejściowe, np. piksele obrazu czy cechy tekstu.
- Warstwy Ukryte: Przetwarzają dane wejściowe, ucząc się abstrakcyjnych reprezentacji. Mogą ich być dziesiątki, setki lub nawet tysiące.
- Warstwa Wyjściowa: Generuje końcowe wyniki, takie jak klasyfikacje lub predykcje.

Funkcje aktywacji decydują, czy dany neuron zostanie aktywowany, przekształcając sumę wejść neuronu w wyjście. Popularne funkcje to:

- ReLU (Rectified Linear Unit): $f(x) = \max(0, x)$
- Sigmoid: $f(x) = 1 / (1 + \exp(-x))$
- Tanh (Hyperbolic Tangent): $f(x) = \tanh(x)$

Wartości na wykresach strat i dokładności:

- Mała wartość straty walidacji - zbiór dobrze dobiera parametry do modelu
- Mała wartość straty trenowania - zbiór dobrze się uczy wzorców ze zbioru

- Duża wartość dokładności walidacji – model ma lepszą zdolność do generalizacji i tym samym oznacza, to że ma dobrze dopasowane parametry modelu.
- Duża wartość dokładności treningowej – model jest lepiej nauczony na podstawie danych treningowych

Model 1 - bez mechanizmu regularyzacji

Do budowy sieci neuronowej zastosowano architekturę składającą się z naprzemiennych warstw Conv2D (z funkcją aktywacji relu) i MaxPooling2D.

Rozpoczęto od map wejściowych o rozmiarach 150x150 (która jest dowolną wartością), a tuż przed warstwą Flatten skończono na mapach o rozmiarze 7x7.

W tej sieci głębokość map cech wzrasta (od 32 do 128), a ich wymiary maleją (od 148x148 do 7x7). Sytuacja taka ma miejsce w prawie wszystkich konwolucyjnych sieciach neuronowych.

Niezbędne okazało się rozwiązanie problemu klasyfikacji binarnej, a więc na końcu sieci umieszczono jedną warstwę Dense o rozmiarze równym 1 oraz funkcję aktywacji sigmoid. Dzięki tej metodzie, stało się możliwe sklasyfikowanie analizowanego obrazu do jednej z klas, poprzez generację wartości prawdopodobieństwa.

Model: "sequential_6"

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_10 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_13 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_11 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_14 (Conv2D)	(None, 34, 34, 128)	73,856
max_pooling2d_12 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_15 (Conv2D)	(None, 15, 15, 128)	147,584
max_pooling2d_13 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten_5 (Flatten)	(None, 6272)	0
dense_12 (Dense)	(None, 512)	3,211,776
dense_13 (Dense)	(None, 1)	513

Total params: 3,453,121 (13.17 MB)
Trainable params: 3,453,121 (13.17 MB)
Non-trainable params: 0 (0.00 B)

Rys. 2. Podgląd modelu sieci neuronowej

Podczas kompilacji modelu skorzystano z funkcji straty w postaci binarnej entropii krzyżowej oraz optymalizatora ‘adam’.

Wstępna obróbka danych

Dane przed przekazaniem do wejść sieci należało odpowiednio sformatować, czyli przedstawić w formie tensorów wartości zmiennoprzecinkowych. Lista poczynionych kroków podczas formatowania:

1. Wczytanie plików obrazów
2. Zdekodowanie formatu JPEG do siatki pikseli w formacie RGB.
3. Zapisanie danych w formie tensorów liczb zmiennoprzecinkowych.
4. Przeskalowanie wartości pikseli z zakresu 0–255 do zakresu [0, 1], ponieważ sieci neuronowe lepiej pracują z małymi wartościami wejściowymi.

W celu wykonania powyższych czynności skorzystano z gotowej biblioteki *ImageDataGenerator* z pakietu *keras.preprocessing.image*.

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

Rys. 3. Rezultat wyszukiwania dla zbioru trenującego i walidacyjnego

Podgląd wartości generatorów

```
kształt danych wsadowych: (20, 150, 150, 3)
kształt etykiet danych wsadowych: (20,)
```

Rys. 4. Podgląd wartości jednego z generatorów

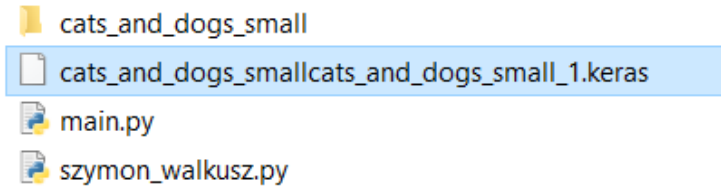
Jak widać, wsad obrazów RGB o wymiarach 150x150 o kształcie (20, 150, 150, 3) i binarne etykiety o kształcie (20,). W każdym wsadzie znajduje się 20 próbek.

Trenowanie modelu

Korzystając z metody ***fit*** przekazano obiekty generatorów trenowania i walidacji oraz określono liczbe epok.

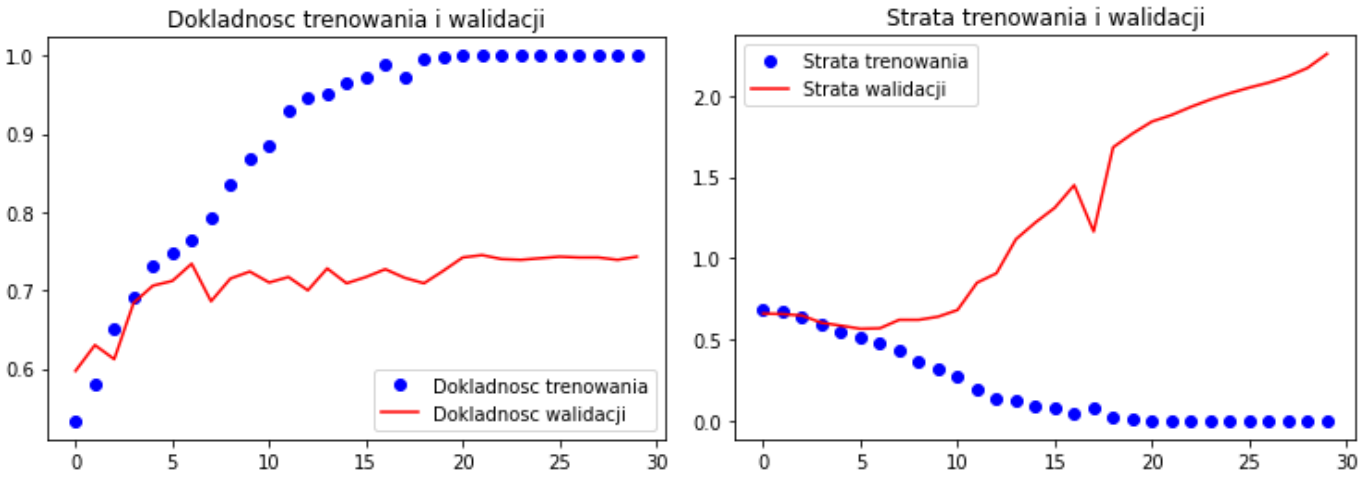
```
Epoch 1/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 20s 200ms/step - acc: 0.5207 - loss: 0.6890 - val_acc:
0.6170 - val_loss: 0.6723
Epoch 2/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 19s 191ms/step - acc: 0.5563 - loss: 0.6825 - val_acc:
0.6150 - val_loss: 0.6498
Epoch 3/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 22s 217ms/step - acc: 0.6038 - loss: 0.6469 - val_acc:
0.6080 - val_loss: 0.6525
Epoch 4/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 21s 209ms/step - acc: 0.6430 - loss: 0.6261 - val_acc:
0.6650 - val_loss: 0.6139
```

Rys. 5. Podgląd kolejnych epok trenowania



Rys. 6. Rezultat zapisania modelu do pliku

Wykresy straty i dokładności pracy modelu podczas przetwarzania danych treningowych i walidacyjnych



Z powyższych wykresów oraz z faktu niezbyt licznego zbioru danych uczących, wynika że sieć ulega nadmiernemu dopasowaniu, co można ograniczyć np. poprzez dropout lub regularyzację L2 lub augmentację danych. Gdy wykresy trenowania i walidacji znacząco odbiegają od siebie, wówczas może to oznaczać przeuczenie sieci z powodu zbyt ubogiego zbioru treningowego. Po zaledwie 5

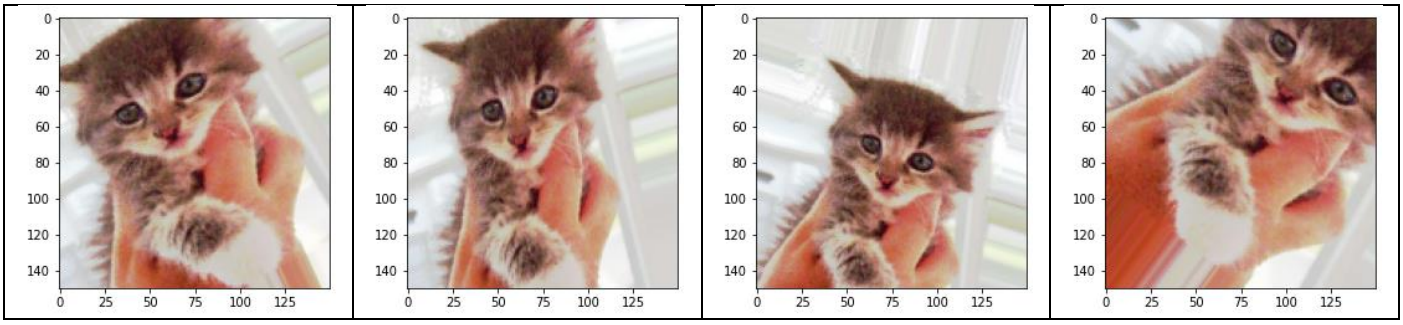
epokach dokładność walidacji zyskuje stabilność na poziomie ok 71%, natomiast w dalszych epokach model uczy się danych na pamięć.

Model 2 - Augmentacja danych

Nadmierne dopasowanie wynika ze zbyt małej liczby próbek, na których model może się uczyć. Model nie może w takiej sytuacji utworzyć uogólnień, które sprawdzą się podczas przetwarzania nowych danych. Gdyby istniał nieskończenie wielki zbiór danych treningowych, to na model działałby każdy możliwy aspekt rozkładu danych, więc nigdy nie uległby przeuczeniu.

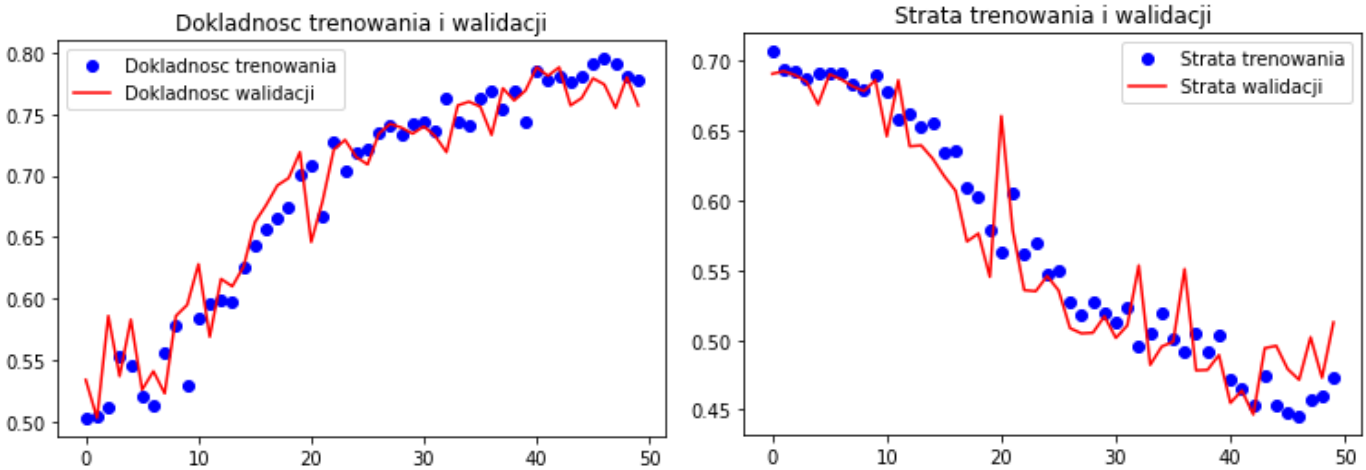
Augmentacja danych to technika generowania większej liczby elementów treningowego zbioru danych poprzez powiększenie liczby próbek na drodze losowych przekształceń zwracających obrazy, które wyglądają wiarygodnie. Przekształceniami tymi mogą być m.in. obrót, skalowanie, przycięcie, odbicie.

Tabela 1. Przykład kilku zmodyfikowanych obrazków



Dzięki augmentacji, wygenerowane są te same dane, jednakże w innej postaci, dzięki czemu model może zwiększyć swoją efektywność na małych zbiorach treningowych, jak również na dużych. Nie daje to jednak gwarancji całkowitego wyeliminowania przeuczenia, ale może je ograniczyć.

Bardzo pomocną metodą przy redukcji nadmiernego dopasowania, jest Dropout. Technika ta losowo wyłącza część neuronów podczas treningu, co zmusza model do bardziej ogólnych reprezentacji danych.



Dzięki augmentacji danych zjawisko przeuczenia sieci, niemalże zostało zażegnane o czym świadczy współbieżność wykresów walidacji i trenowania. Model do 50 epoki sukcesywnie się uczył, osiągając dokładność na poziomie ok 77%.

Model 3 - Wykorzystywanie wytrenowanej sieci CNN w zadaniu binarnej klasyfikacji obrazów przy użyciu architektury VGG16

Możliwe są dwie techniki stosowania wytrenowanej wcześniej sieci: ekstrakcja cech lub dostrajanie.

Ekstrakcja cech w przypadku sieci konwolucyjnej polega na przepuszczeniu nowych danych przez konwolucyjną bazę wcześniej wytrenowanej sieci i wytrenowaniu nowego klasyfikatora na wyjściu tej sieci. Zwykle pierwsze warstwy biorą udział w ekstrakcji, ponieważ określają bardziej ogólny stan modelu i nie zagłębiają się w dane reprezentujące drobne szczegóły ostatnich warstw.

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_layer_9 (InputLayer)	(None, 150, 150, 3)	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1,792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36,928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73,856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147,584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295,168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590,080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590,080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0

Total params: 14,714,688 (56.13 MB)
Trainable params: 14,714,688 (56.13 MB)
Non-trainable params: 0 (0.00 B)

Rys. 7. Model VGG16

Ostatecznie powstała mapa cech o kształcie (4, 4, 512) z której może korzystać klasyfikator.

Postanowiono rozszerzyć model conv_base dodając do niego warstwy Dense i używać całej tej konstrukcji w celu przetworzenia danych wejściowych.

Rozwiązanie to umożliwia korzystanie z augmentacji danych, ponieważ każdy obraz wejściowy przechodzi przez konwolucyjną bazę za każdym razem, gdy jest analizowany przez model. Technika ta jest bardziej kosztowna obliczeniowo.

Dodanie modelu conv_base do modelu sekwencyjnego

Model: "functional_447"

Layer (type)	Output Shape	Param #
input_layer_9 (InputLayer)	(None, 150, 150, 3)	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1,792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36,928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73,856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147,584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295,168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590,080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590,080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0
flatten (Flatten)	(None, 8192)	0
dense_18 (Dense)	(None, 256)	2,097,408
dense_19 (Dense)	(None, 1)	257

Total params: 16,812,353 (64.13 MB)
Trainable params: 16,812,353 (64.13 MB)
Non-trainable params: 0 (0.00 B)

Rys. 8. Model sekwencyjny

Konwolucyjna baza sieci VGG16 ma 14 714 688 parametrów. Klasyfikator dodawany do tej bazy ma ponad 2 miliony parametrów co wynika z działania:

$$16{,}812{,}353 - 14{,}714{,}688 = 2\ 097\ 665$$

Przed kompilacją i trenowaniem modelu zamrożono bazę konwolucyjną.

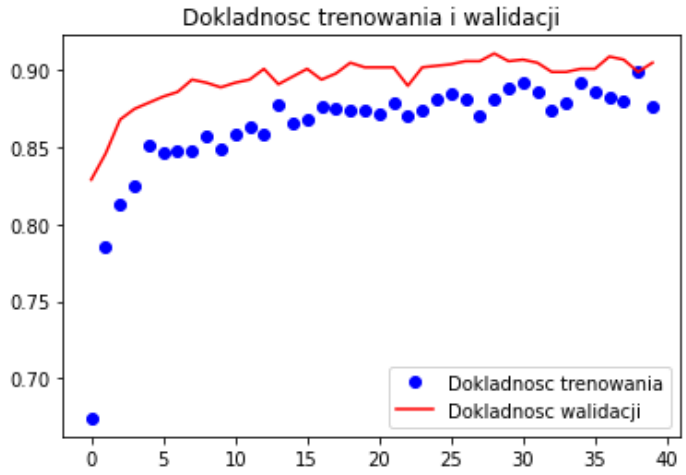
Zamrażanie warstwy lub zestawu warstw polega na zapobieganiu aktualizacji ich wag w procesie trenowania. Dzięki temu, reprezentacje wyuczone wcześniej przez bazę konwolucyjną, nie są modyfikowane podczas trenowania.

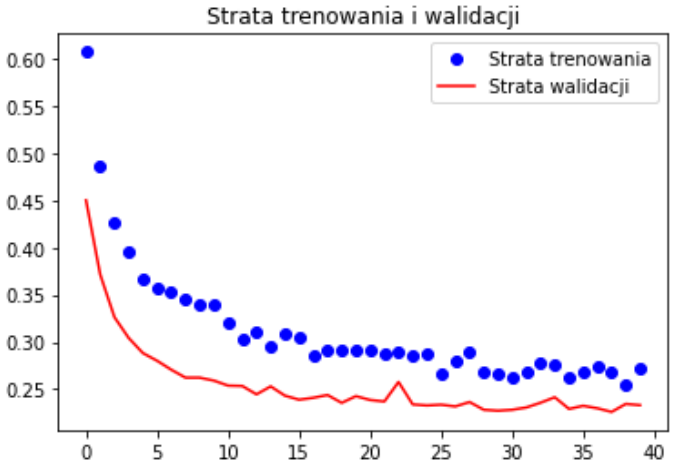
```
Liczba tensorów wag poddawanych trenowaniu przed zamrożeniem bazy: 30
Liczba tensorów wag poddawanych trenowaniu po zamrożeniu bazy: 4
```

Rys. 9. Porównanie liczby tensorów

Przy tej konfiguracji trenowane zostały wagi z dwóch warstw Dense, stanowiące cztery tensory wag po dwa tensory na warstwę (główna macierz wag i wektor wartości progowych).

Następnie, w celu wprowadzenia zmian, skompilowano model, po czym model był gotowy do trenowania przy użyciu augmentacji danych.





Uzyskana dokładność walidacji sięga około 90% i to o wiele lepszy wynik od tego, który był uzyskany przy użyciu małej sieci konwolucyjnej trenowanej od podstaw. Wykresy trenowania i walidacji są ze sobą względnie zbieżne, co dowodzi, że model nie doznał przeuczenia.

Dostrajanie sieci

Dostrajanie sieci polega na odmrażaniu kilku górnych warstw zamrożonej bazy modelu używanej do ekstrakcji cech i trenowaniu jej łącznie z nową częścią modelu (w niniejszym przypadku tą częścią modelu jest w pełni połączony klasyfikator).

Proces ten określamy mianem dostrajania, ponieważ modyfikuje on częściowo wytrenowane wcześniej bardziej abstrakcyjne reprezentacje modelu w celu dostosowania ich do bieżącego problemu.

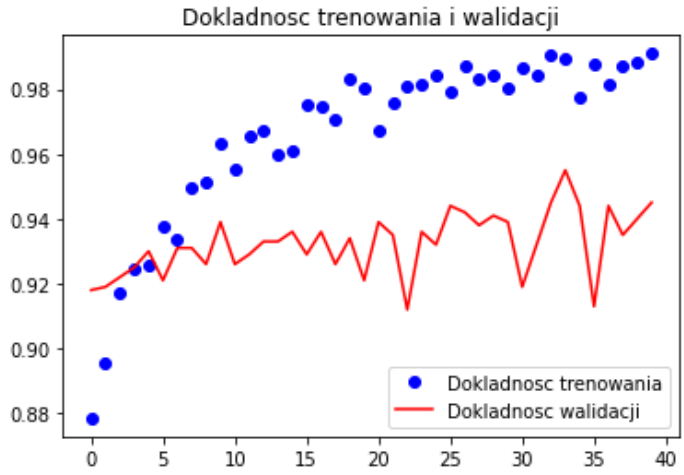
W poprzednich krokach zamrożono bazę konwolucji modelu VGG16, aby wytrenować losowo zainicjowany klasyfikator kończący sieć. Z tego powodu możliwe jest tylko dostrajanie górnych warstw konwolucyjnej bazy po wytrenowaniu klasyfikatora.

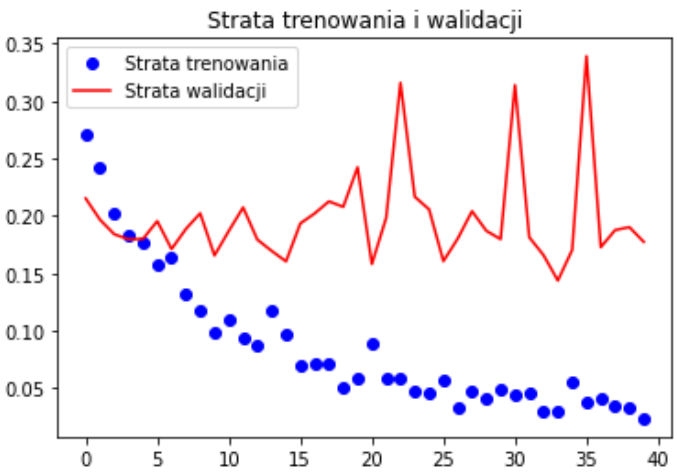
Dostrajanie sieci należy przeprowadzać w następujący sposób:

1. Dodać zaprojektowaną sieć gęstą do końca bazy wytrenowanego już modelu.
2. Zamrozić bazę sieci.
3. Wytrenować nową część sieci.
4. Odmrozić niektóre warstwy bazy sieci.
5. Wytrenować razem: odmrożone warstwy oraz nową część sieci

Pierwsze 3 kroki wykonano podczas ekstrakcji cech.

Aby uniknąć nadmiernego uczenia i zredukować czas trenowania modelu, odmrożono ostatnie 3 warstwy, które zawierają mniej danych do przetworzenia, niż pierwsze warstwy.





Poprzez zastosowanie techniki dostrajania, model osiągnął dokładność trenowania na poziomie ok 98% i 94% dla walidacji. Pomimo wrażenia pozornej rozbieżności wykresów funkcji trenowania i walidacji, należy zwrócić uwagę na stosunek wartości tych funkcji względem siebie. Relatywnie te funkcje mają zbliżone wartości, więc model został dobrze dostrojony i nie uległ drastycznemu przeuczeniu, o ile można je stwierdzić. Są to bardzo pożądane wyniki i bardzo zadawalające.

Sprawdzenie na danych testowych

```
Found 1000 images belonging to 2 classes.
50/50 ━━━━━━━━━━━━━━━━━━━━ 35s 705ms/step - acc: 0.9448 - loss: 0.2450
dokładnosc podczas testowania: 0.9350000023841858
```

Rys. 10. Rezultat testu na danych testowych

Jak widać, dokładność na nowych danych osiągnęła ok 94%. Jest to satysfakcjonujący wynik.

Podsumowanie

Korzystając z małych zbiorów danych do trenowania sieci neuronowej, można uzyskać wysokie wartości predykcji dla klasyfikacji obrazów. Aby uzyskać taki efekt, niezbędna jest augmentacja danych do sztucznego generowania nowych danych, dobór odpowiedniej architektury oraz dostrajanie. Skuteczność pierwotnego modelu zwiększono ok 33% w stosunku do jego pierwotnej wartości, a tym samym o 23% bezwzględnie:

$$(71 + 71*33\%)\% \approx 94\%$$

Powyższe badania dowodzą, że poprzez zastosowanie właściwego modelu można uzyskać praktyczną użyteczność z dokładnością do 94%.