**Sztuczna Inteligencja 2020/21**

**Uczenie maszynowe – sieci neuronowe (laboratorium).**

**Błażej Bałdyga, 02.06.2021**

Do uczenia maszynowego metodą sieci neuronowych wybrano dataset **MNIST**, który zawiera dużą bazę obrazów cyfr zapisanych pismem ręcznym. Obrazy tam zawarte są podzielone na zbiór testowy i treningowy. Zbiór testowy zawiera **60 000**obrazów, a zbiór treningowy zawiera **10 000** obrazów. Obrazy w bazie zapisane są w postaci tensorów:

28 x 28 = 784

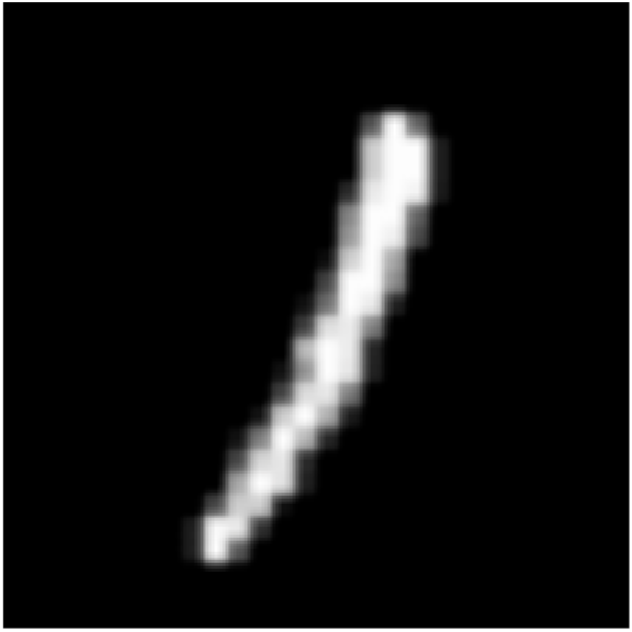
Piksele w tensorze reprezentowane są liczbami 0...255 (odcienie szarości). Poniżej wklejono 5 przykładowych obrazów z datasetu, otrzymanych za pomocą kodu:



**no of sample = 55**



**no of sample = 100**



**no of sample = 140**



**no of sample = 180**



**no of sample = 200**



**no of sample = 240**

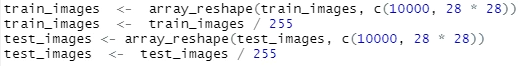


Do bazy dołączono wektor etykiet zawierający etykiety w postaci cyfr 0...9 oznaczających cyfrę zapisaną ręcznie w danym obrazie bazy.

Ze względu na ograniczone możliwości techniczne ograniczamy liczbę próbek w zbiorze treningowym do 10000 za pomocą kodu:



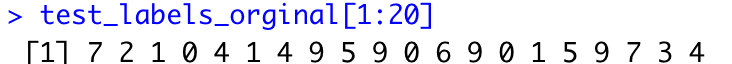
Do uczenia sieci neuronowej potrzebujemy wektorów, stąd zmniejszamy rank oryginalnego tensora z (10000, 28, 28) na (10000, 784), gdzie 10000 to liczba próbek, a 784 to liczba składowych wektora uczącego. Dodatkowo wskazane jest aby wartości poszczególnych składowych (pikseli) wektora uczącego zawierały się w przedziale [0, 1] co uzyskujemy dzieląc oryginalne wartości przez 255. Odpowiedni kod pokazano poniżej:

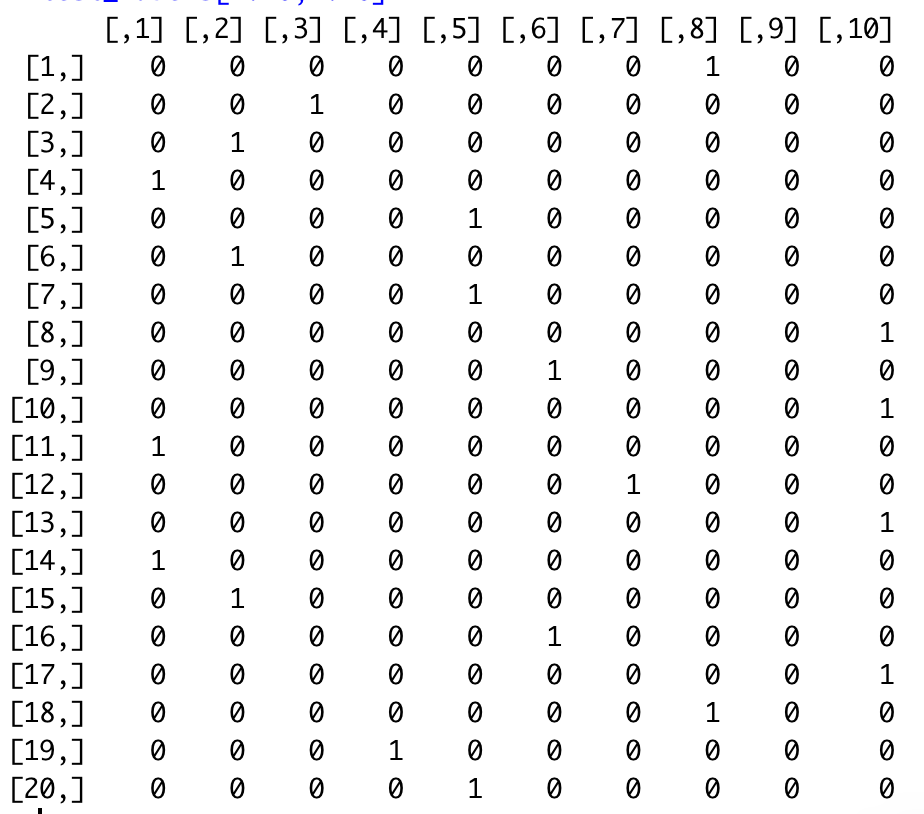


W celu zakwalifikowania obrazka do jednej z 10-ciu klas (0, 1, 2,...,9) w ostatniej warstwie umieszczono 10 jednostek („perceptronów”), gdzie każda jednostka będzie rozpoznawała określoną cyfrę. W związku z tym należy przekształcić oryginalny wektor etykiet (labels) do postaci tablicy [10000, 10], gdzie w każdym wierszu występuje dokładnie jedna jedynka - w kolumnie odpowiadającej jednostce rozpoznającej daną cyfrę. Wykorzystano do tego kod:

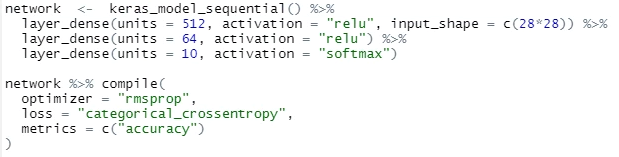


Poniżej przedstawiono 20 początkowych składowych oryginalnego wektora oraz „te same” składowe po przekształceniu „to\_categorical”:





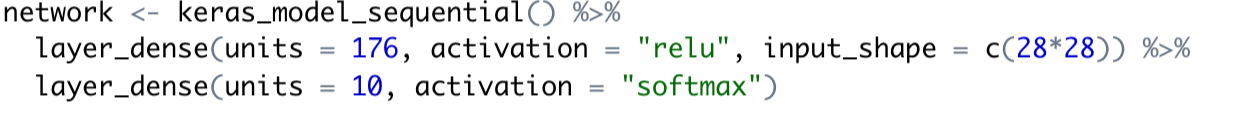
Dane zostały przygotowane do procesu tworzenia modelu sieci neuronowej. W następnym etapie tworzymy model sieci neuronowej oraz dokonujemy jego kompilacji. Przykładowy kod modelu z dwiema warstwami ukrytymi pokazano poniżej:



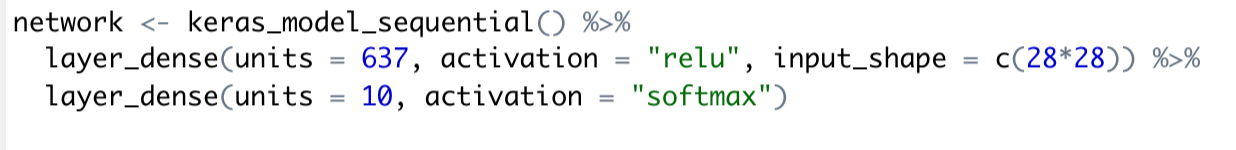
W naszych modelach wszystkie warstwy oprócz ostatniej wykorzystują funkcję aktywacji „relu”, natomiast ostatnia warstwa wykorzystuje funkcję „softmax”, gdyż chcemy otrzymać wyniki z przedziału [0,1]. Parametry kompilacji modelu dobrano kierując się tym, że celem naszym jest klasyfikacja na kategorie.

W celu przetestowania i porównania różnych sieci utworzono 5 modeli o parametrach podanych poniżej (2 modele z jedną warstwą ukrytą, 2 modele z dwiema warstwami, 1 model z trzema warstwami ukrytymi) :

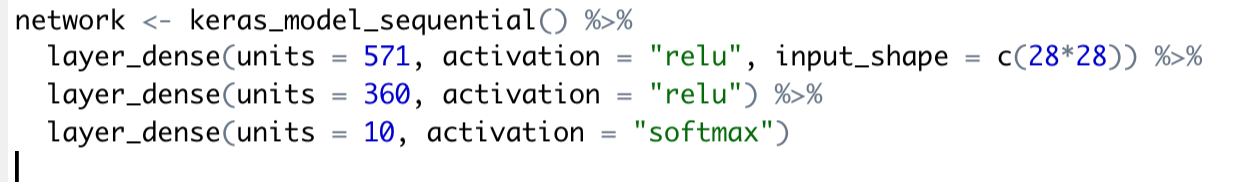
Model 1



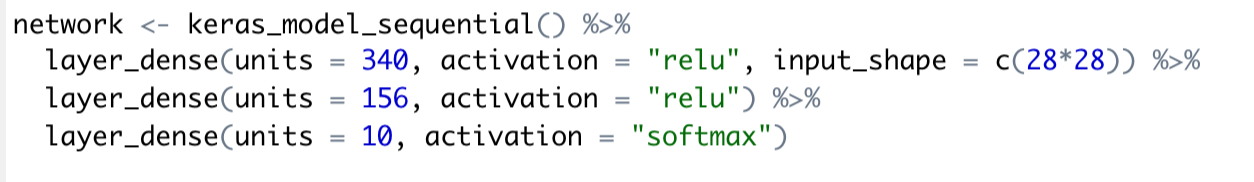
Model 2



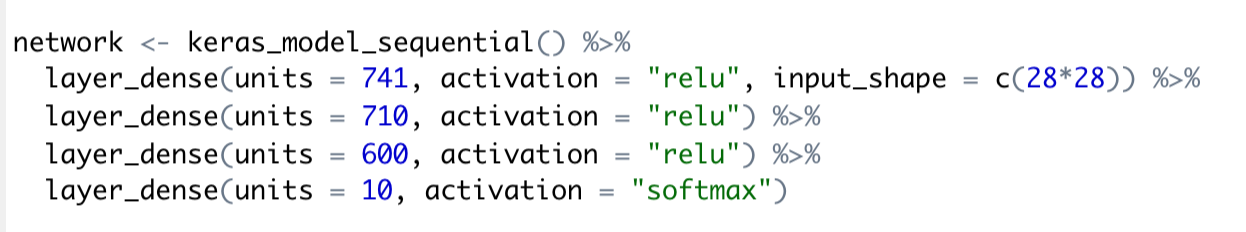
Model 3



Model 4



Model 5



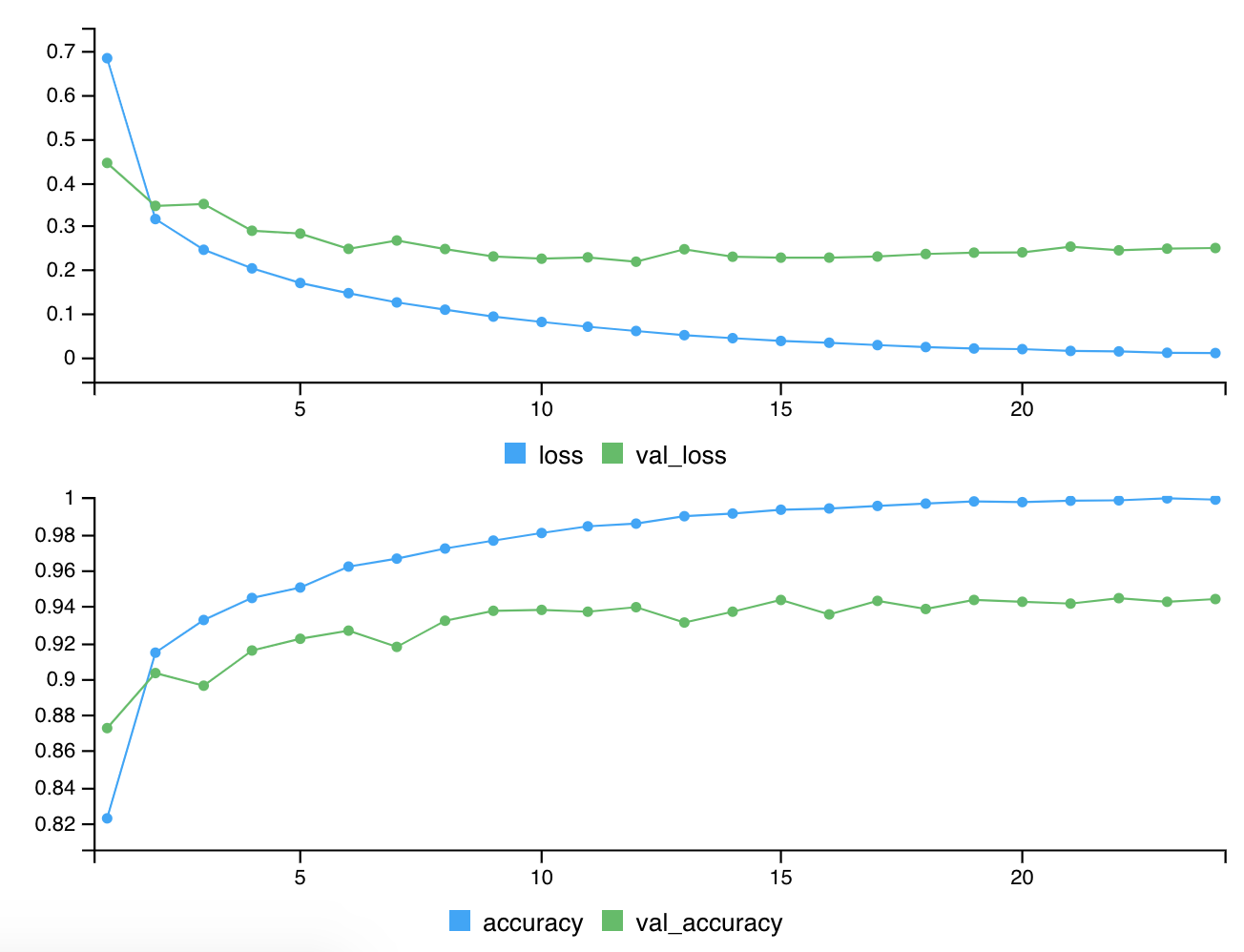
Następnie trenujemy modele za pomocą kodu:



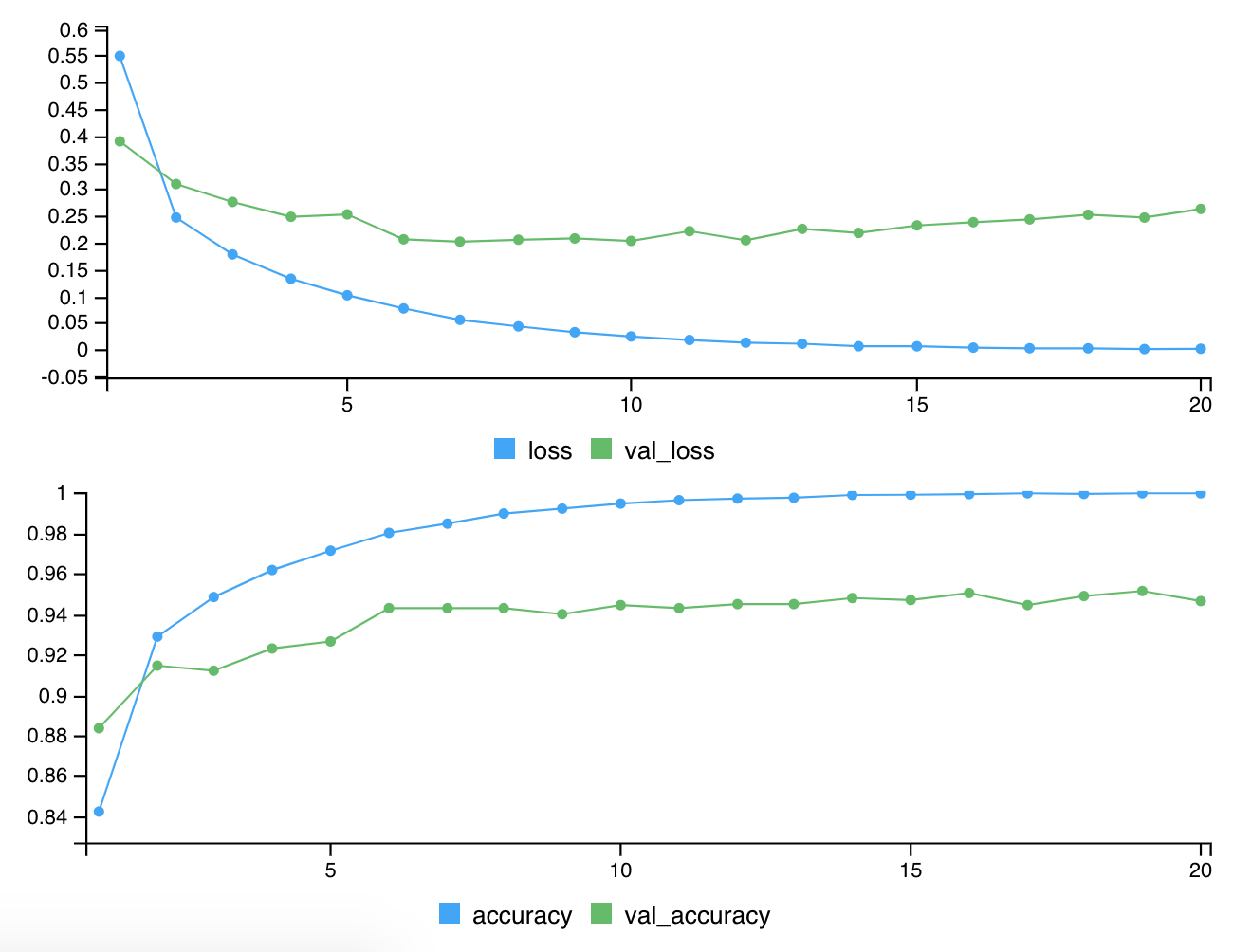
Rezerwujemy tutaj 20% zbioru treningowego do walidacji procesu uczenia. Hiperparametry epochs i batch\_size możemy modyfikować w razie potrzeby.

Poniżej przedstawiono wykresy błędu uczenia (loss) oraz dokładność rozpoznawania (accuracy) w trakcie uczenia dla części treningowej i walidacyjnej dla pięciu zaproponaowanych modeli.

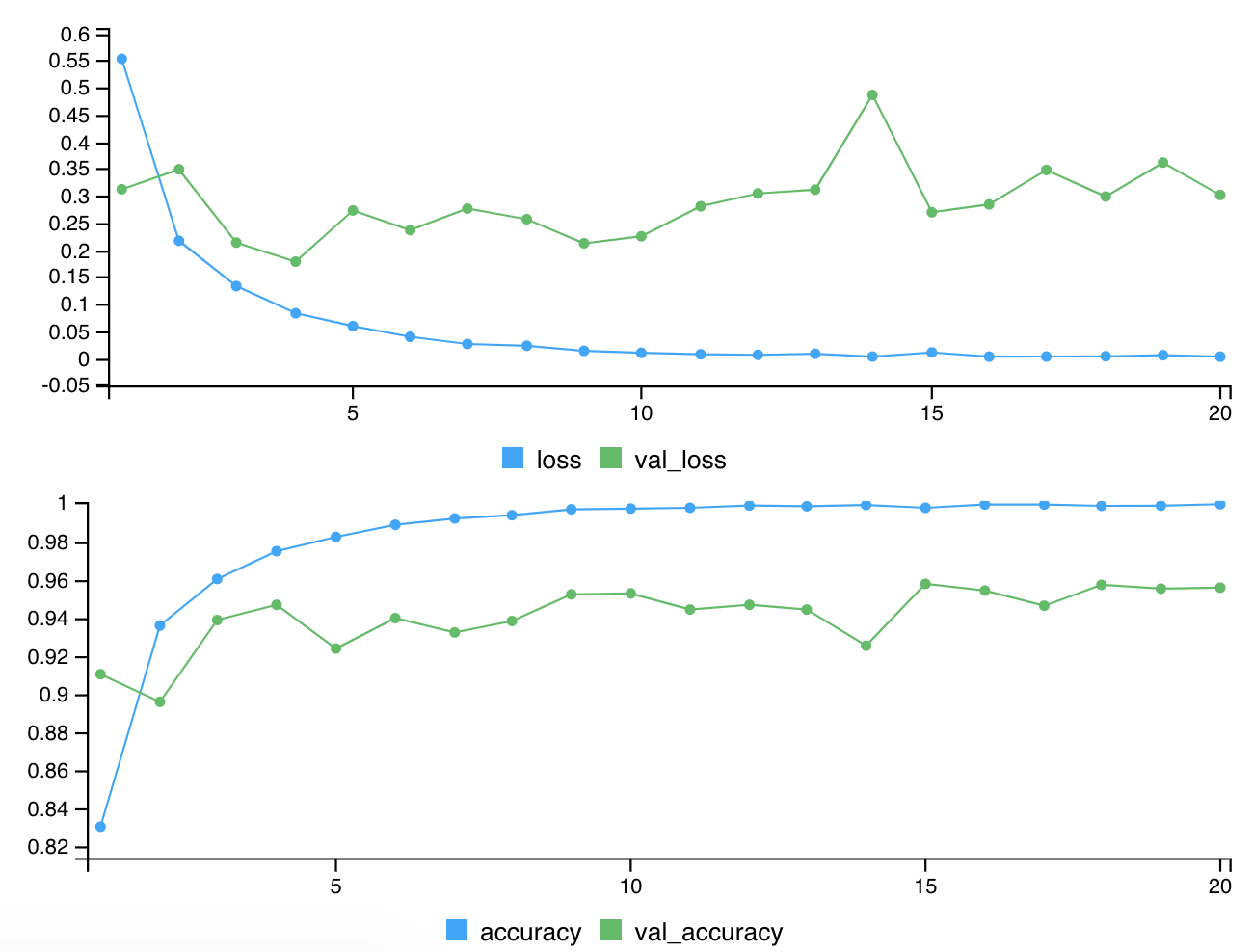
Model 1



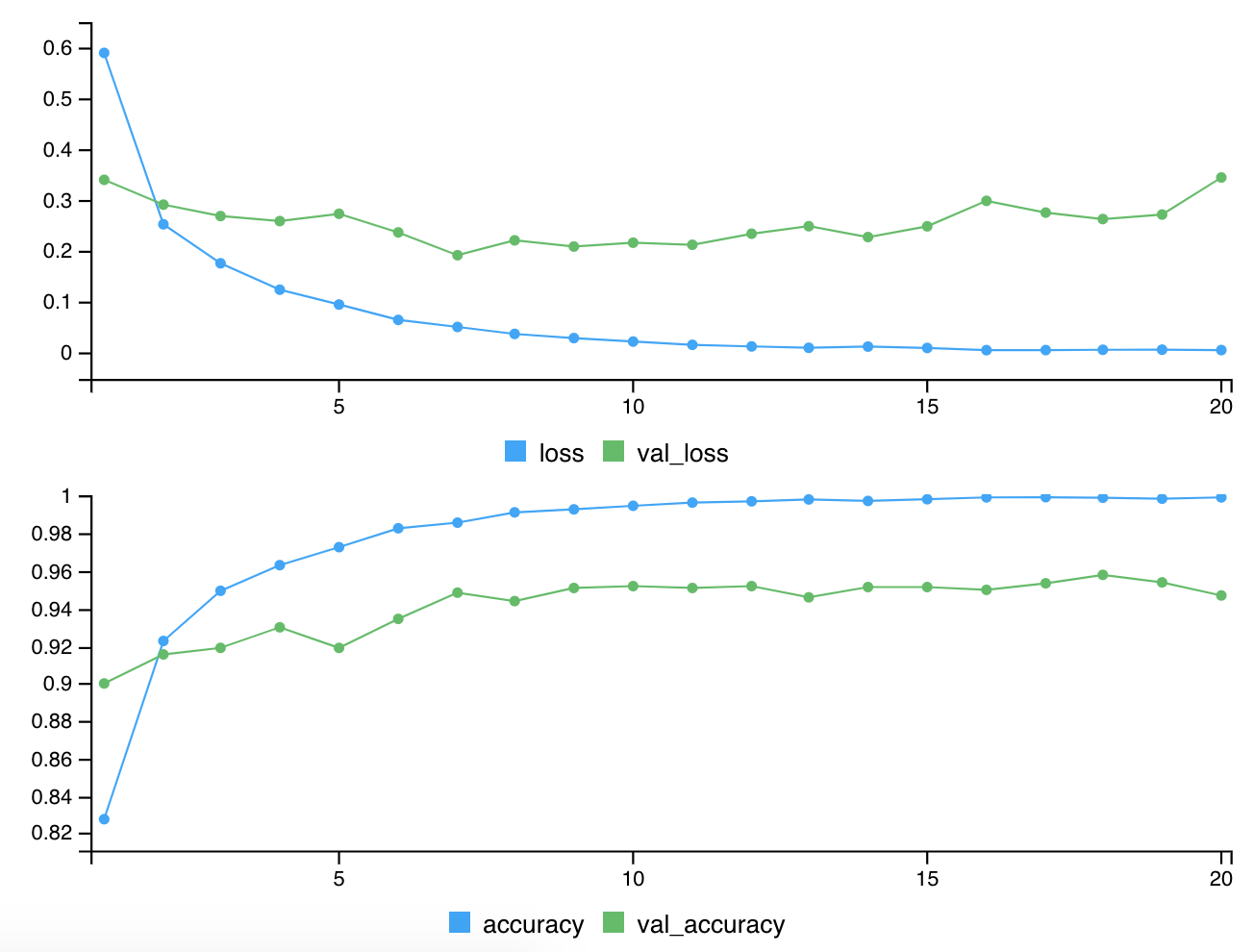
Model 2



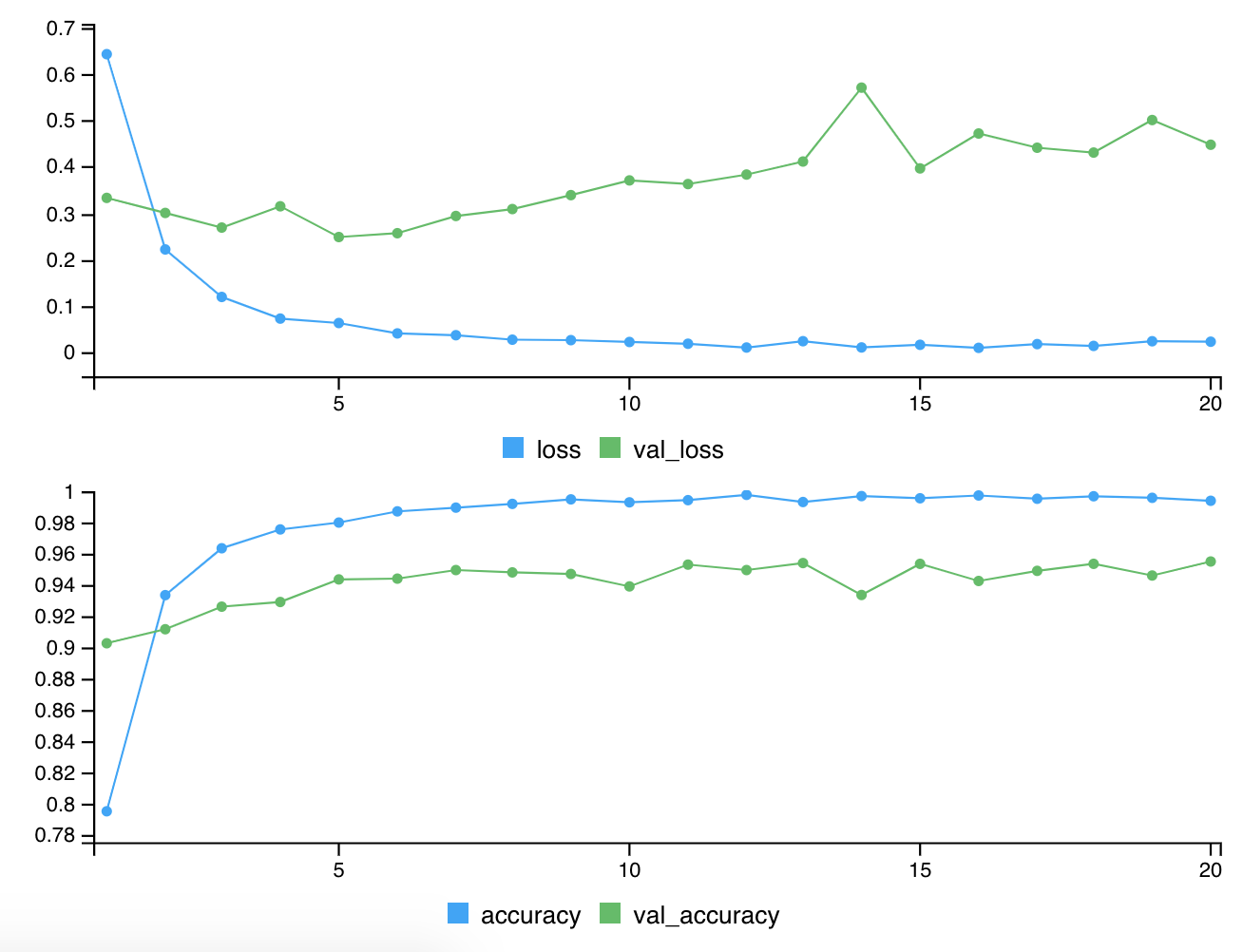
Model 3



Model 4



Model 5



Na podstawie wykresów zdecydowano się na ustalenie liczby epochs dla badanych modeli zgodnie z tabelką:

|  |  |
| --- | --- |
| **Model nr** | **epochs** |
| **Model 1** | 20 |
| **Model 2** | 24 |
| **Model 3** | 20 |
| **Model 4** | 20 |
| **Model 5** | 20 |

Następnie dokonujemy ewaluacji zaproponowanych modeli na zbiorze testowym za pomocą kodu:



Wyniki przedstawiono w poniższej tabeli:

|  |  |  |
| --- | --- | --- |
| **Model nr** | **loss** | **accuracy** |
| **Model 1** | 0.2073029 | 0.9495000 |
| **Model 2** | 0.2069484 | 0.9587000 |
| **Model 3** | 0.247978 | 0.961200 |
| **Model 4** | 0.2936268 | 0.9477000 |
| **Model 5** | 0.3290846 | 0.9586000 |

Na tej podstawie zdecydowano, że najlepszym z badanych modeli jest Model **2**, dla którego accuracy wynosi **0.9587 a loss 0.2069484** .

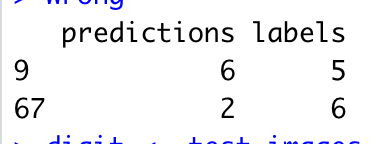
Zobaczmy, jak w przykładowych szczegółach prezentują się nasze wyniki. Dokonajmy predykcji i porównajmy wyniki z oryginalnymi etykietami reprezentującymi poprawne wartości dla 100 pierwszych liter:



Następnie sprawdźmy, które litery z tych 100 zostały źle sklasyfikowane za pomocą kodu:



Wyniki przedstawiono poniżej:



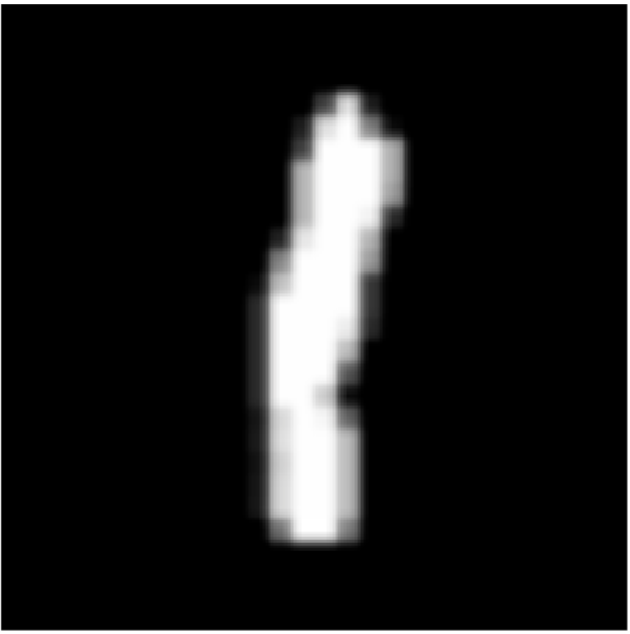
Poniżej pokazano obrazki tych liter, które zostały źle sklasyfikowane:





Dla przykładu poniżej pokazano również trzy obrazki , które zostały prawidłowo rozpoznane:

dla 1



dla 9



dla 7

