Sieć konwolucyjna

Autor: Rafał Behrendt
246643@student.pwr.edu.pl
Prowadząca: dr hab. inż. Urszula Markowska-Kaczmar
3 grudnia 2021

Abstract

Celem ćwiczenia jest zapoznanie się działaniem sieci z warstwami konwolucyjnymi. Eksperymenty przeprowadzono na zbiorze MNIST zawierającym 60000 danych treningowych oraz 10000 danych testowych.

Kod źródłowy znajduje się w poniższym linku:

https://github.com/aspirew/neuralNetworks/tree/list4

1 Eksperymenty

Za domyślne hiperparametry przyjęto:

- Ilość warstw konwolucyjnych: 2
- Ilość warstw w pełni połączonych: 2 (1 warstwa ukryta o rozmiarze 128 neuronów i warstwa wyjściowa)
- Wielkość kernela: (3, 3)
- Wielkości map cech: 32, 64
- Współczynnik uczenia: 0.01
- Inicjalizacja wag: metoda xaviera
- Optymalizator: Adam
- Funkcja straty: Entropia krzyżowa
- Funkcja aktywacji: ReLU

Każdy test przeszedł przez 10 epok zanim został zakończony. Aby zbadać prędkość nauki przyjęto wartość progową 0.98 dokładności i sprawdzono, po ilu epokach wartość ta jest osiągana. Wartość ta określona jest w tabelach w kolumnie z nagłówkiem **wytrenowanie**

2 Porównanie wydajności sieci MLP i sieci z jedną warstwą konwolucyjną

Został przeprowadzony eksperyment porównujący wydajność sieci MLP bez warstw konwolucyjnych z siecią wielowarstwową z jedną warstwą konwolucyjną. Sieć MLP posiadała dwie warstwy ukryte o rozmiarach 128 i 64 neurony. Po każdej warstwie zastosowano dropout. W drugiej sieci znajdowała się jedna warstwa konwolucyjna i domyślne warstwy w pełni połączone.

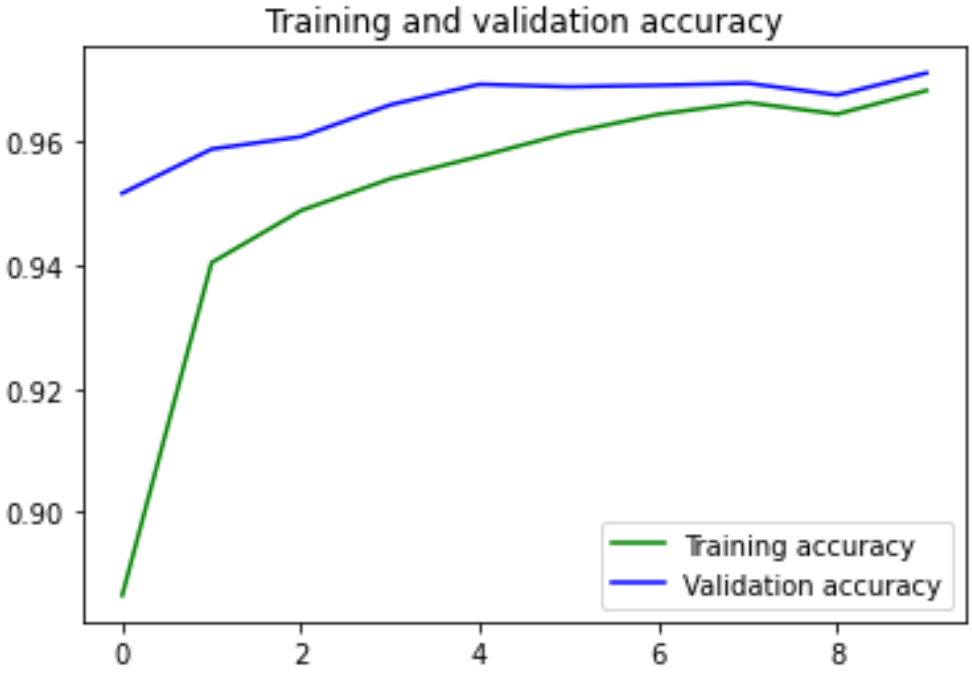
2.1 Sieć MLP

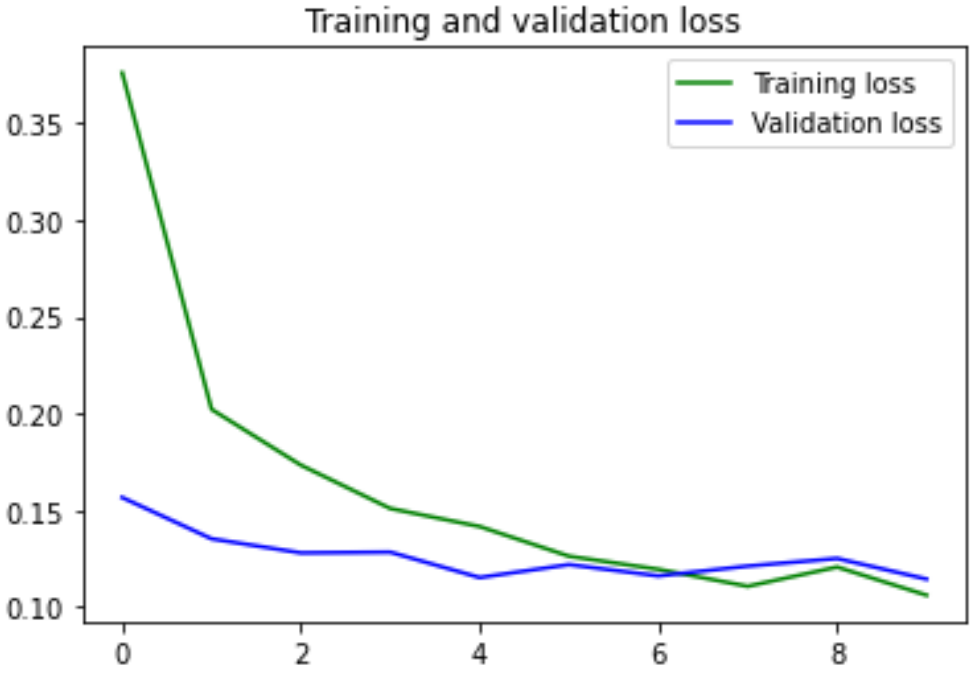
Epoka	Czas wykonania [s]	Błąd	Dokładność	Błąd dev	Dokładność dev
1	3	0.376	0.8865	0.1567	0.9517
2	2	0.2022	0.9404	0.1353	0.9588
3	2	0.1734	0.9489	0.1281	0.9608
4	2	0.151	0.954	0.1284	0.9660
5	2	0.1417	0.9577	0.1153	0.9693
6	2	0.1263	0.9615	0.1220	0.9689
7	2	0.1196	0.9645	0.1162	0.9692
8	2	0.1108	0.9664	0.1212	0.9695
9	2	0.1208	0.9645	0.1252	0.9676
10	2	0.1062	0.9683	0.1145	0.9712

Tablica 1: Wyniki wykonania sieci MLP

Błąd na zbiorze testowym: 0.1127
Dokładność na zbiorze testowym: 0.9717

Sieć osiąga bardzo dobre wyniki nawet bez warstw konwolucyjnych.


Training and validation accuracy
0.96
0.94
0.92
0.90
0
2
4
6
8
Training accuracy
Validation accuracy


Training and validation loss
0.35
0.30
0.25
0.20
0.15
0.10
0
2
4
6
8
Training loss
Validation loss

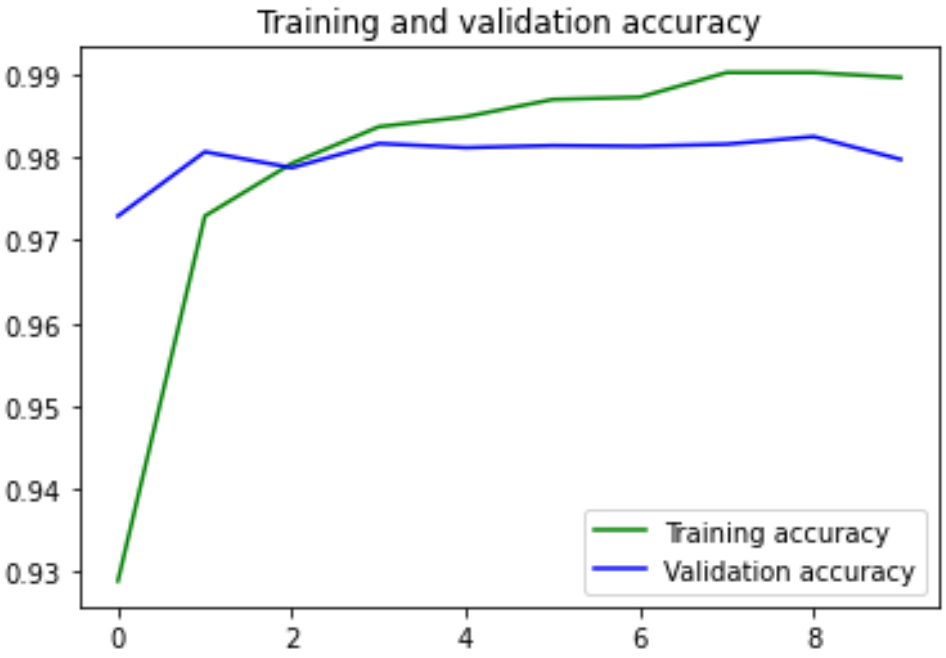
2.2 Sieć z warstwą konwolucyjną

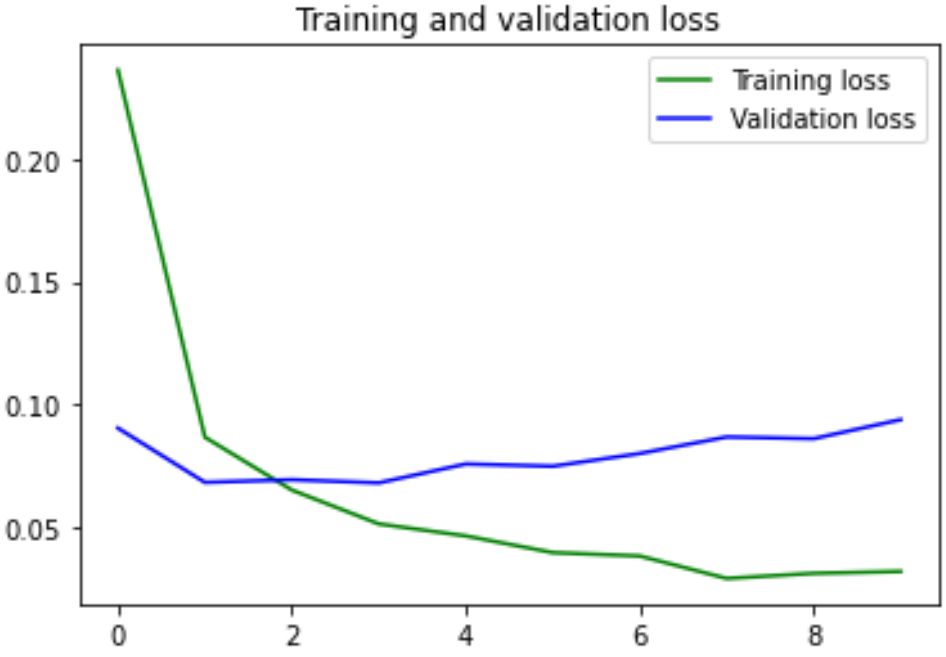
Epoka	Czas wykonania [s]	Błąd	Dokładność	Błąd dev	Dokładność dev
1	11	0.2364	0.9289	0.0902	0.9729
2	11	0.0865	0.9729	0.0680	0.9807
3	11	0.0649	0.9793	0.0691	0.9787
4	11	0.0511	0.9837	0.0678	0.9817
5	11	0.0462	0.9849	0.0755	0.9812
6	11	0.0393	0.9870	0.0746	0.9814
7	11	0.0380	0.9872	0.0789	0.9813
8	11	0.0288	0.9902	0.0866	0.9816
9	10	0.0309	0.9902	0.0859	0.9825
10	11	0.0317	0.9896	0.0936	0.9797

Tablica 2: Wyniki wykonania sieci MLP

Błąd na zbiorze testowym: 0.0682
Dokładność na zbiorze testowym: 0.9829

Sieć osiąga lepsze wyniki, ale to na co należy przede wszystkim zwórcić uwagę to fakt, sieć uczy się dużo szybciej i już po 3 epoce osiąga moment w którym zaczyna się przeuczać, na co wskazuje zwiększająca się wartość błędu dla zbioru walidacyjnego. Sieć MLP po 10 epokach wciąż poprawia swoje działanie. Z drugiej strony sieć MLP pod względem wydajnościowym jest niemal 3 razy szybsza, co też należy wziąć pod uwagę.


Training and validation accuracy
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
2
4
6
8
Training accuracy
Validation accuracy


Training and validation loss
0.20
0.15
0.10
0.05
0
2
4
6
8
Training loss
Validation loss

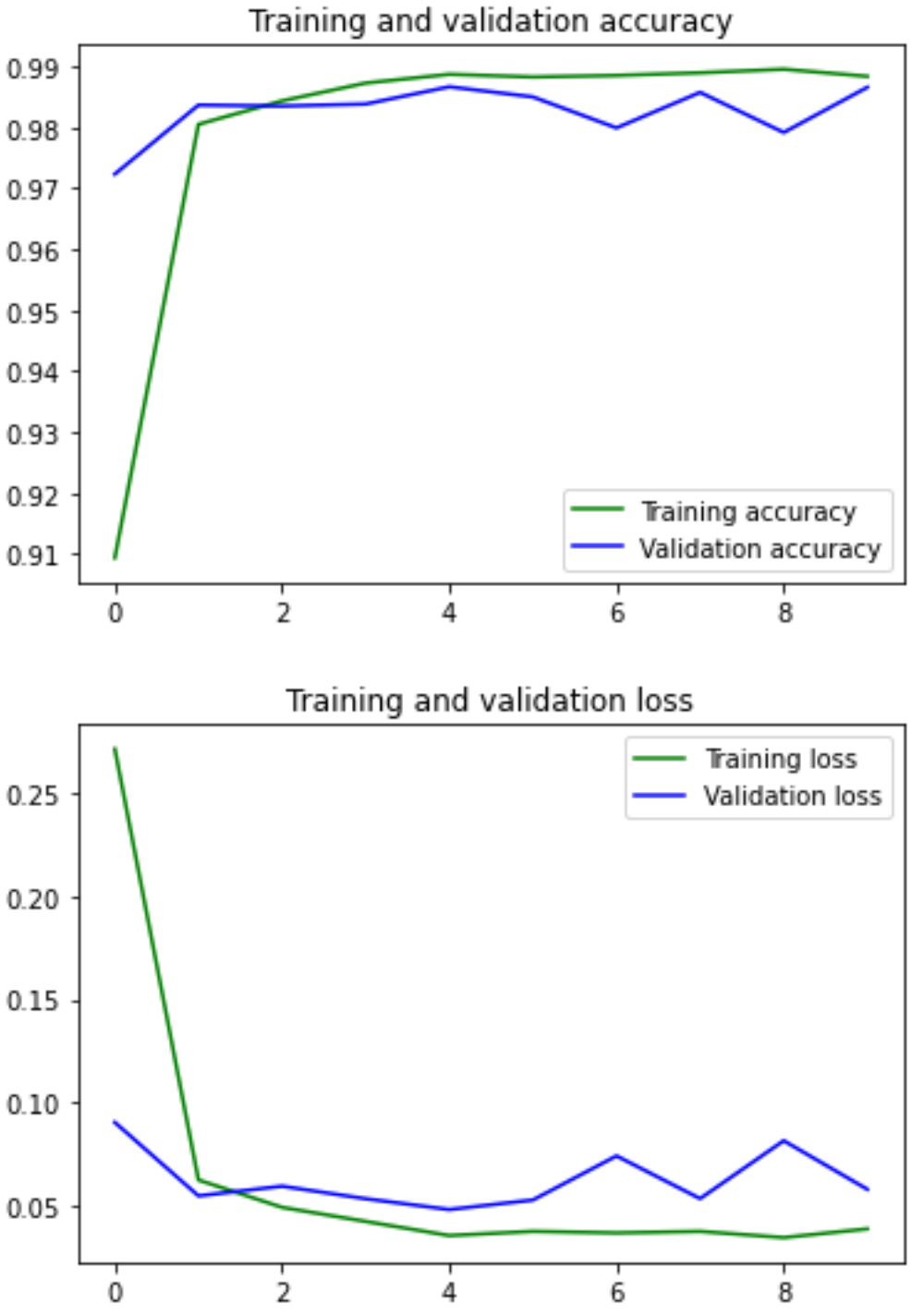
3 Test ilości warstw konwolucyjnych z warstwami poolingu

Zbadano jak ilość warstw konwolucyjnych wpływa na skuteczność sieci. Zastosowano kernel wielkości (3, 3), za wielkość ramki odpowiadał wykorzytany framework keras, ilość map cech wynosiła w pierwszej warstwie 32 i z każdą warstwą zwiększała się dwukrotnie. Po każdej warstwie zastosowano warstwę poolingu. Przedstawione w tabelce wyniki odnoszą się do wyników sieci na zbiorze testowym. Wykresy odnoszą się do wyników sieci na zbiorze treningowym i walidacyjnym.

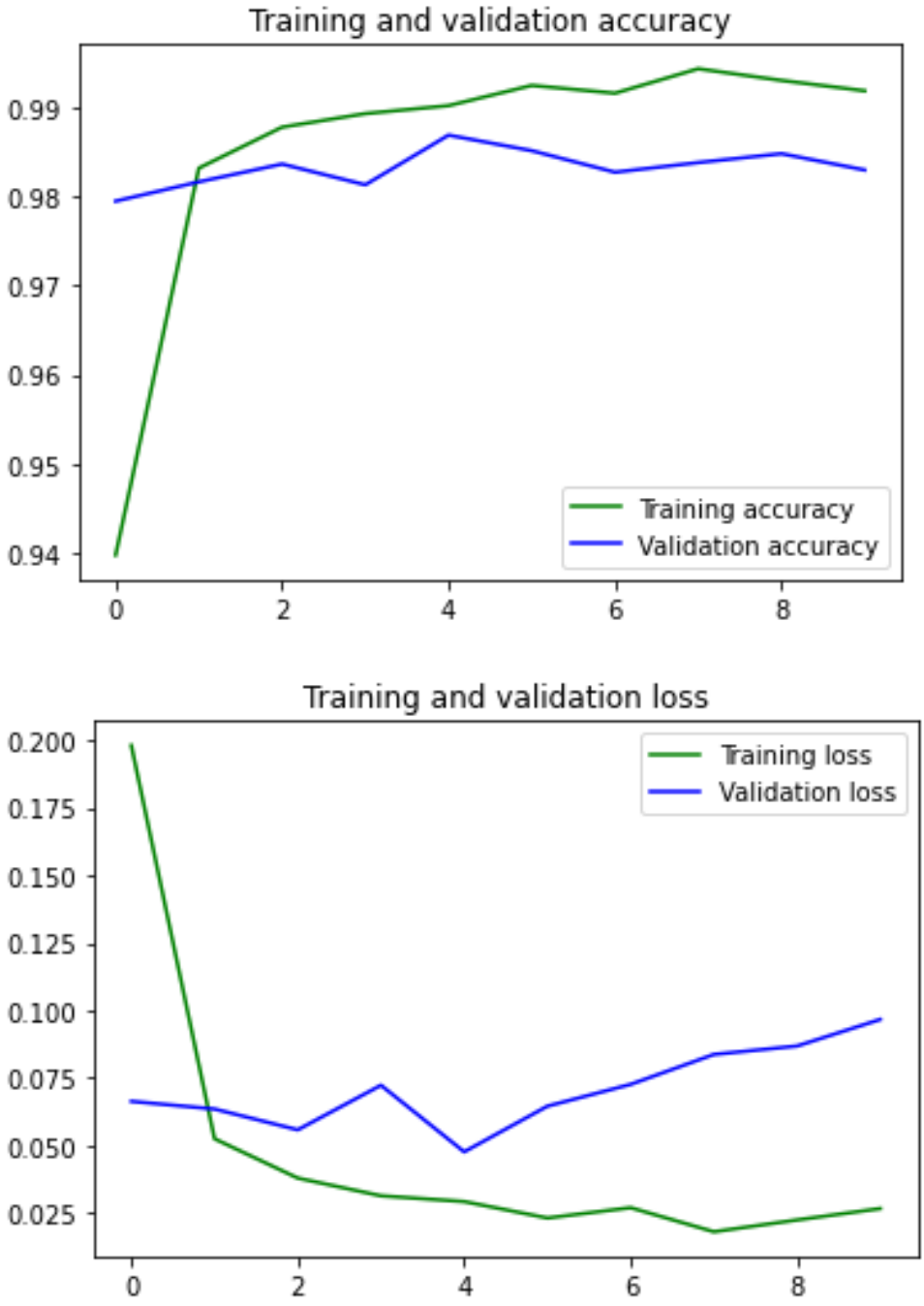
Ilość warstw konwolucyjnych	Czas trenowania [s]	Wytrenowanie [epoka]	Błąd	Skuteczność
3	220	1	0.0524	0.9876
2	170	1	0.0772	0.9850
1	90	1	0.1102	0.9810

Tablica 3: Zastosowanie różnej ilosci warstw konwolucyjnych

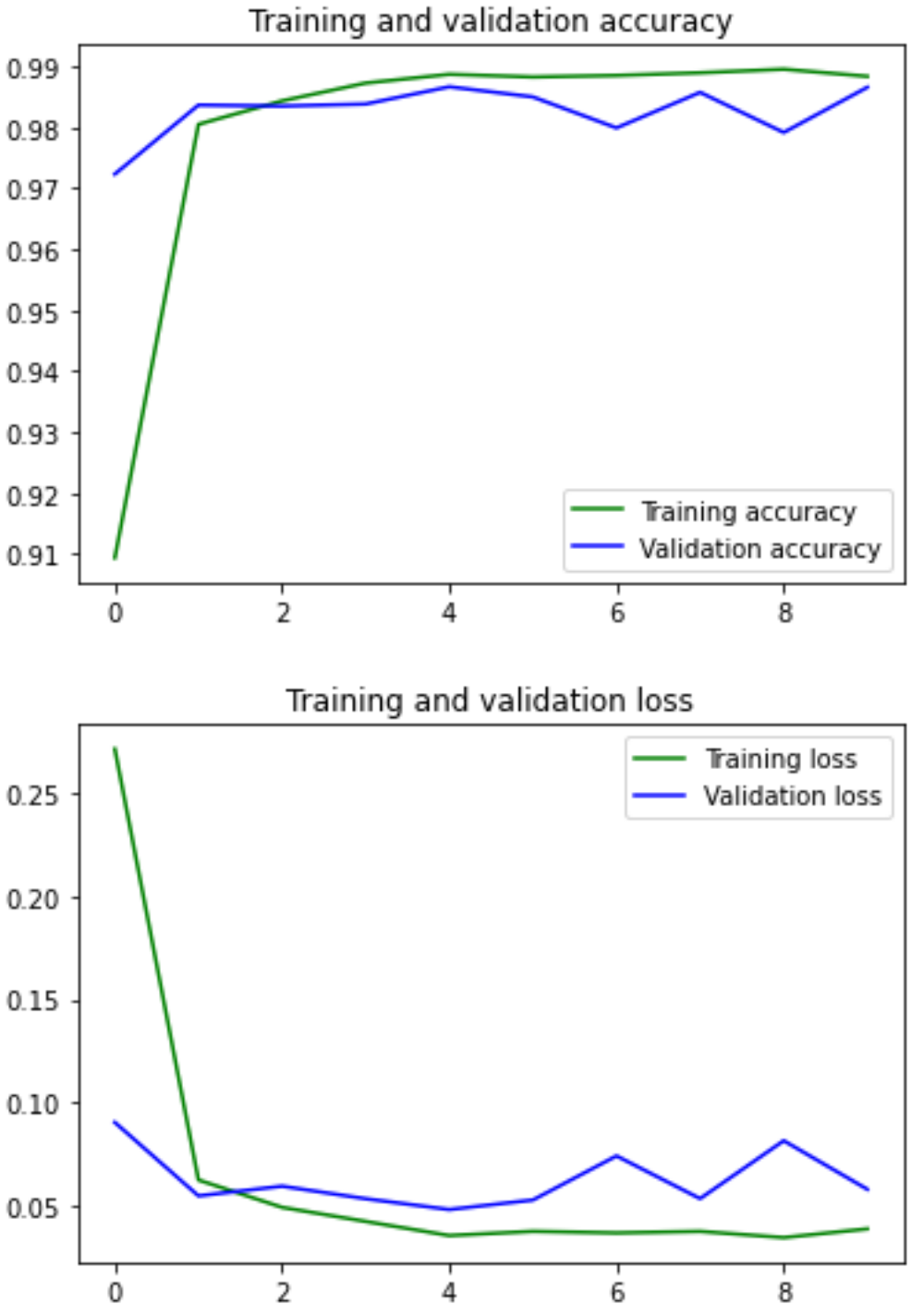
Wraz ze wzrostem ilości warstw konwolucyjnych rośnie dokładność wyników uczenia. Można jednak zauważyć też, że wyniki zbioru walidacyjnego coraz bardziej odbiegają od wyników na zbiorze trenującym co może świadczyć przeuczeniu sieci.



Rysunek 1: 3 warstwy konwolucyje z 3 warstwami poolingu



Rysunek 2: 2 warstwy konwolucyje z 2 warstwami poolingu



Rysunek 3: 1 warstwa konwolucyje z 1 warstwaą poolingu

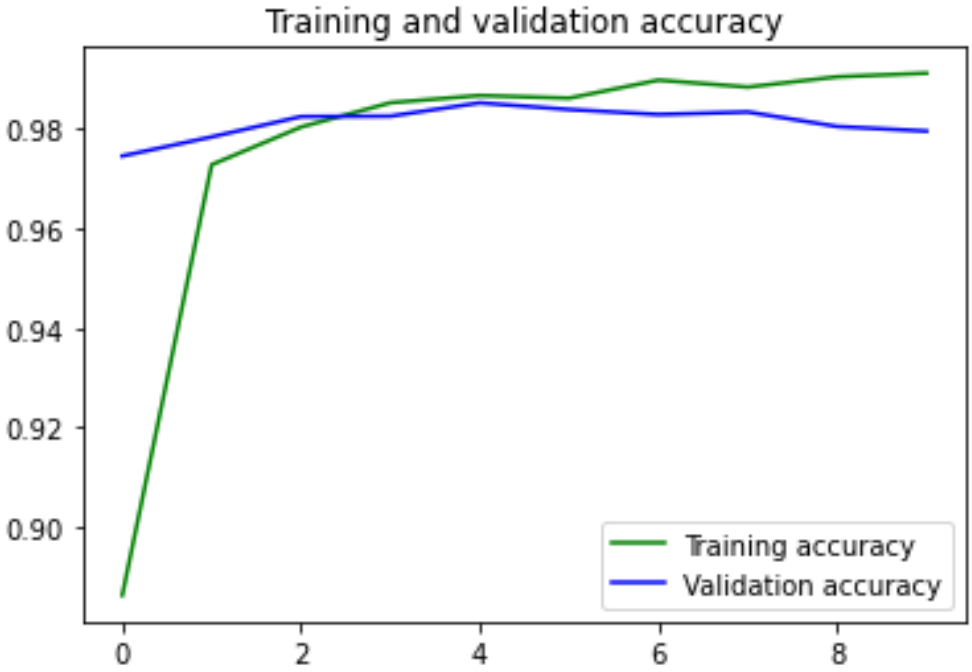
4 Test ilości warstw konwolucyjnych bez warstw poolingu

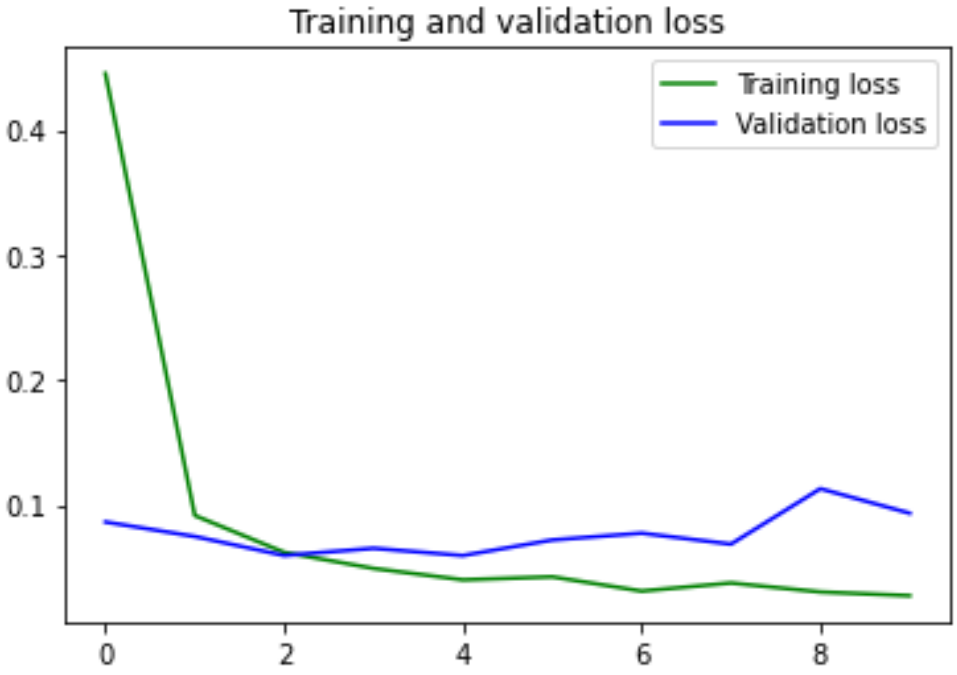
Powtórzono poprzednie badanie bez wykorzystania warstw poolingu.

Ilość warstw konwolucyjnych	Czas trenowania [s]	Wytrenowanie [epoka]	Błąd	Skuteczność
3	1421	2	0.088	0.98
2	497	2	0.0788	0.9831
1	111	2	0.1156	0.9782

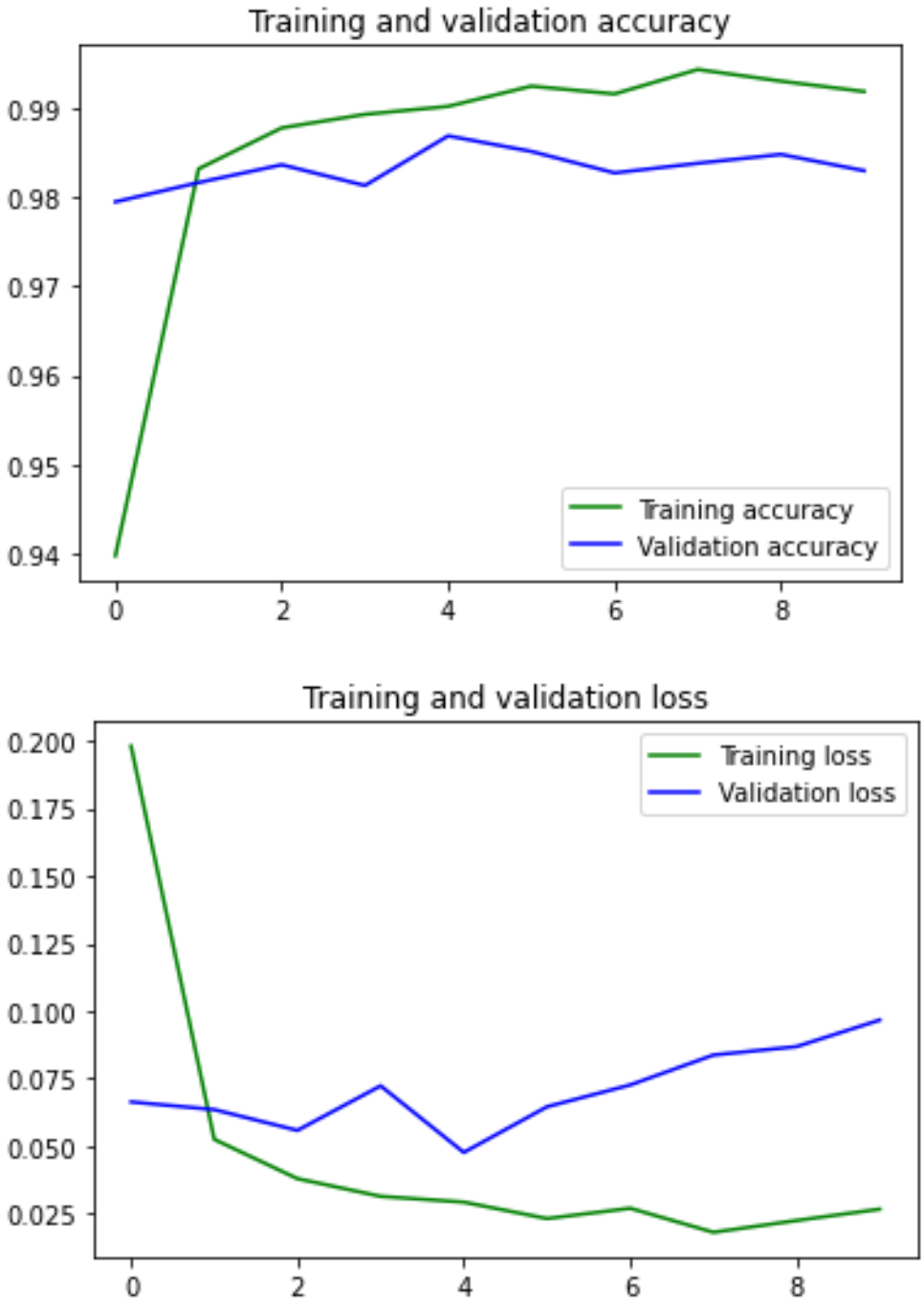
Tablica 4: Zastosowanie różnej ilosci warstw konwolucyjnych bez warstw poolingu

Można zauważyć dużą różnicę w porównaniu z poprzednim eksperymentem. Zastosowanie poolingu ma na celu zmniejszenie wielkosci obrazu, czym samym zmniejszamy ilość koniecznych obliczeń, a także ignorujemy nieistotne dla terningu cechy. Jak można zauważyć za dużo szczegółów branych pod uwagę znacząco wydłuża wykonanie sieci pod względem wydajnościowym jak i pod względem prędkości zmiany błędu. Ostateczne wyniki skuteczności nie odbiegają od eksperymentu z wykorzystaniem warstw poolingu, są nawet nieco gorsze.

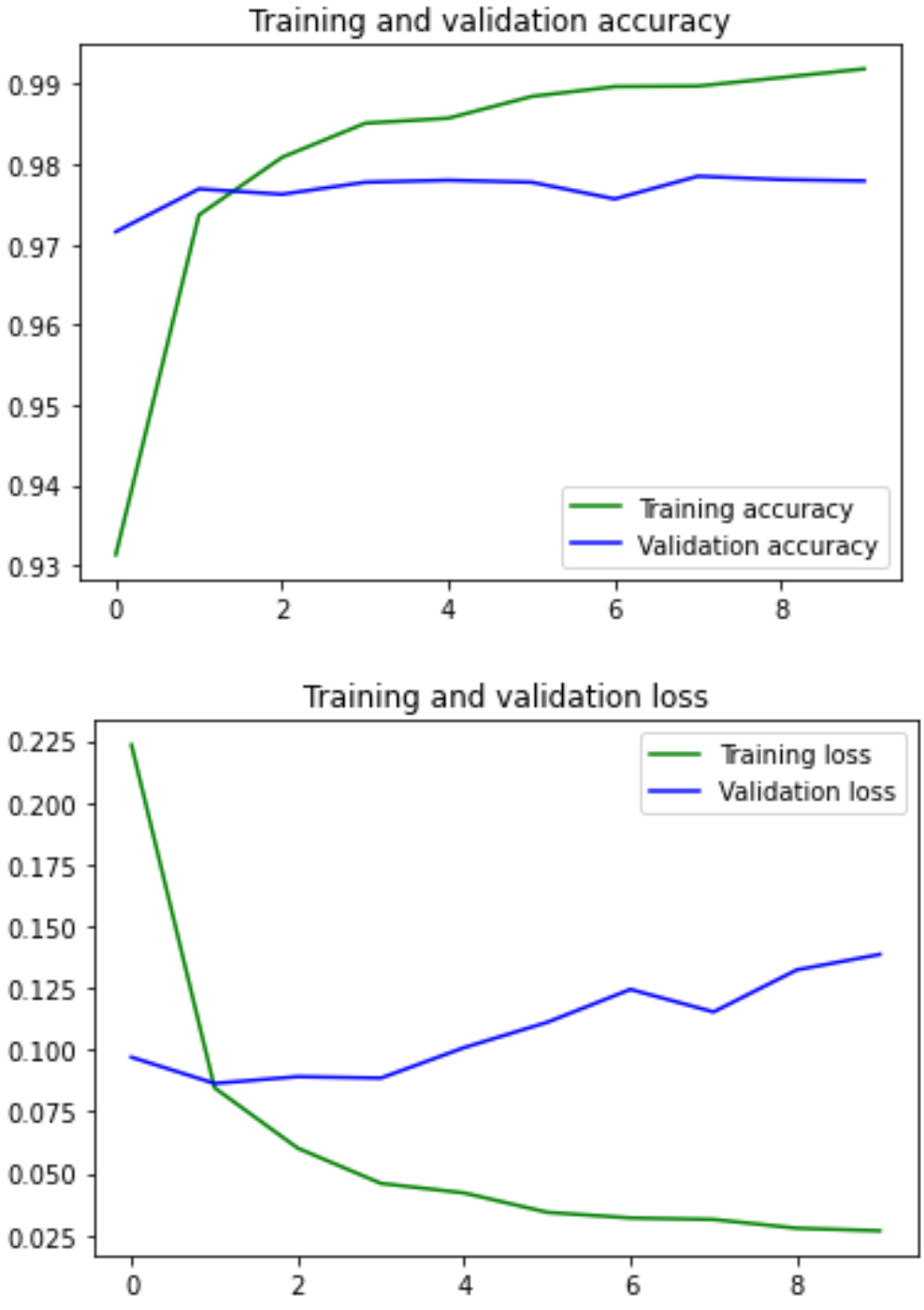




Rysunek 4: 3 warstwy konwolucyje z 3 warstwami poolingu



Rysunek 5: 2 warstwy konwolucyje z 2 warstwami poolingu



Rysunek 6: 1 warstwa konwolucyje z 1 warstwą poolingu

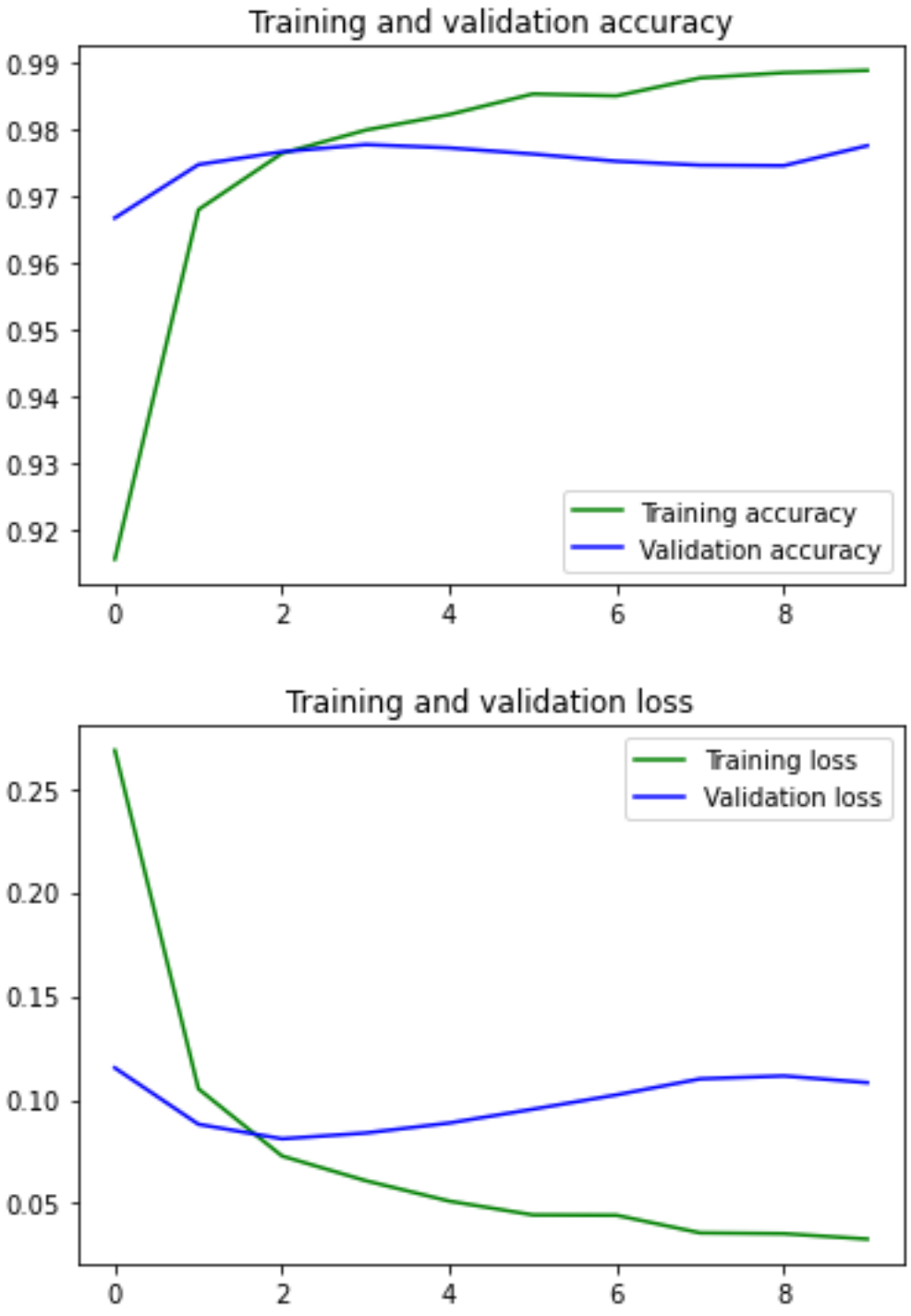
5 Test wielkości kernela

Zbadano jak wielkość kernela wpływa na skuteczność i prędkosć nauki sieci. Badanie przeprowadzono na jednej warstwie konwolucyjnej. Przedstawione w tabelce wyniki odnoszą się do wyników sieci na zbiorze testowym. Wykresy odnoszą się do wyników sieci na zbiorze treningowym i walidacyjnym.

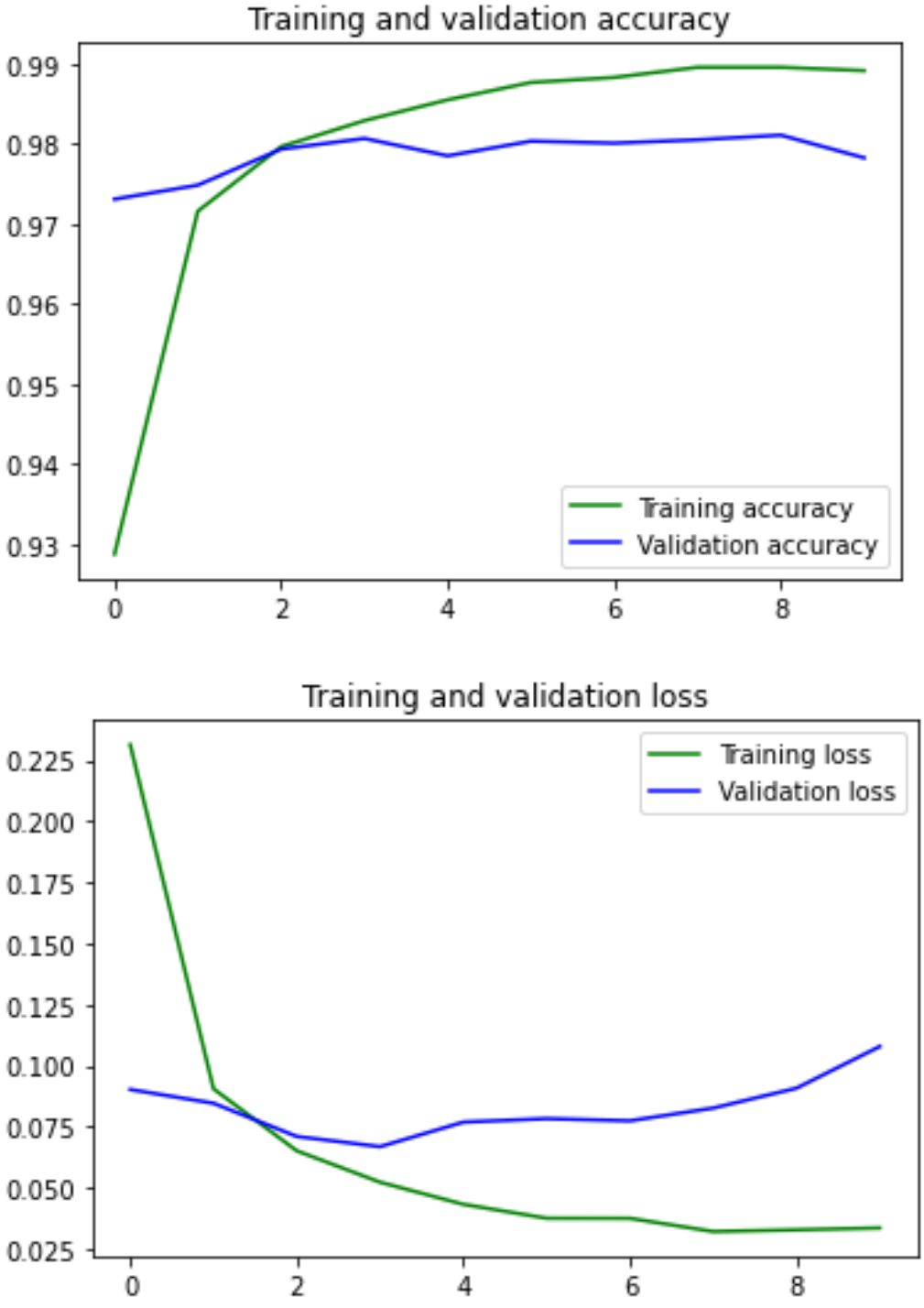
Wielkość kernela	Czas trenowania [s]	Osiągnięcie progu [epoka]	Błąd	Skuteczność
(2, 2)	84	4	0.0873	0.9801
(3, 3)	108	2	0.0764	0.9811
(5, 5)	122	3	0.0744	0.9850
(7, 7)	147	3	0.0744	0.9810
(11, 11)	220	4	0.0864	0.9818

Tablica 5: Zastosowanie różnej wielkości kernela

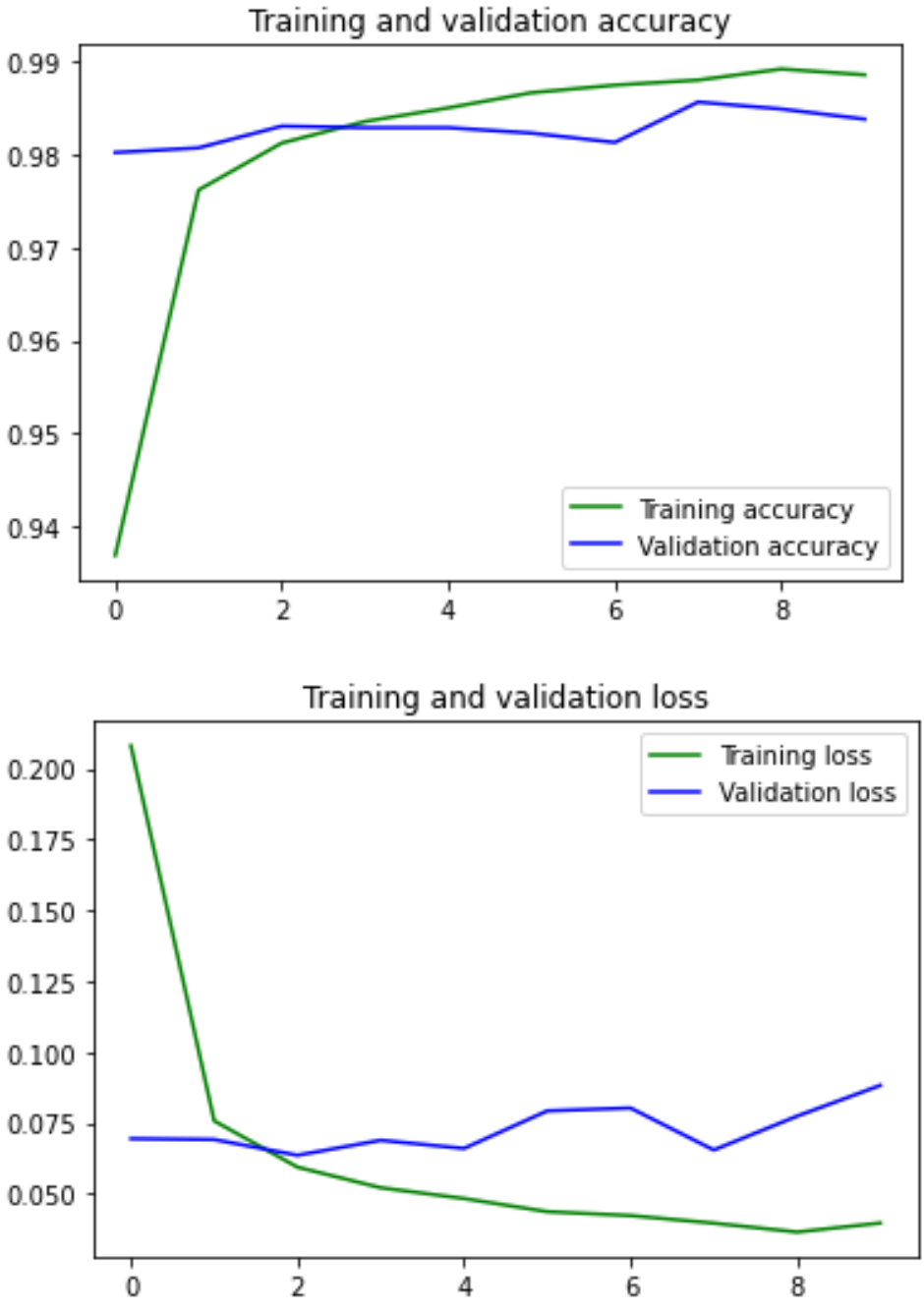
Można zauważyć że dopasowanie wielkości kernela ma znaczenie dla skuteczności sieci. Zbyt niski kernel może okazać się za mały aby zauważyć daną istotną cechę. Zbyt duży z kolei może brać pod uwagę zbyt ogólny obszar obrazu, czym samym przeoczyć cechy które faktycznie są istotne. Dobór wielkości kernela powinien zależeć od wielkości obrazu. Dla problemu w tym zadaniu, można zauważyć, że dla wielkości kernela 3 sieć uczy się najszybciej. Nie widać znaczącej różnicy w skuteczności. Ponadto wraz ze wzrostem kernela, rośnie czas obliczeń, ponieważ sieć ma więcej wag do dopasowania.



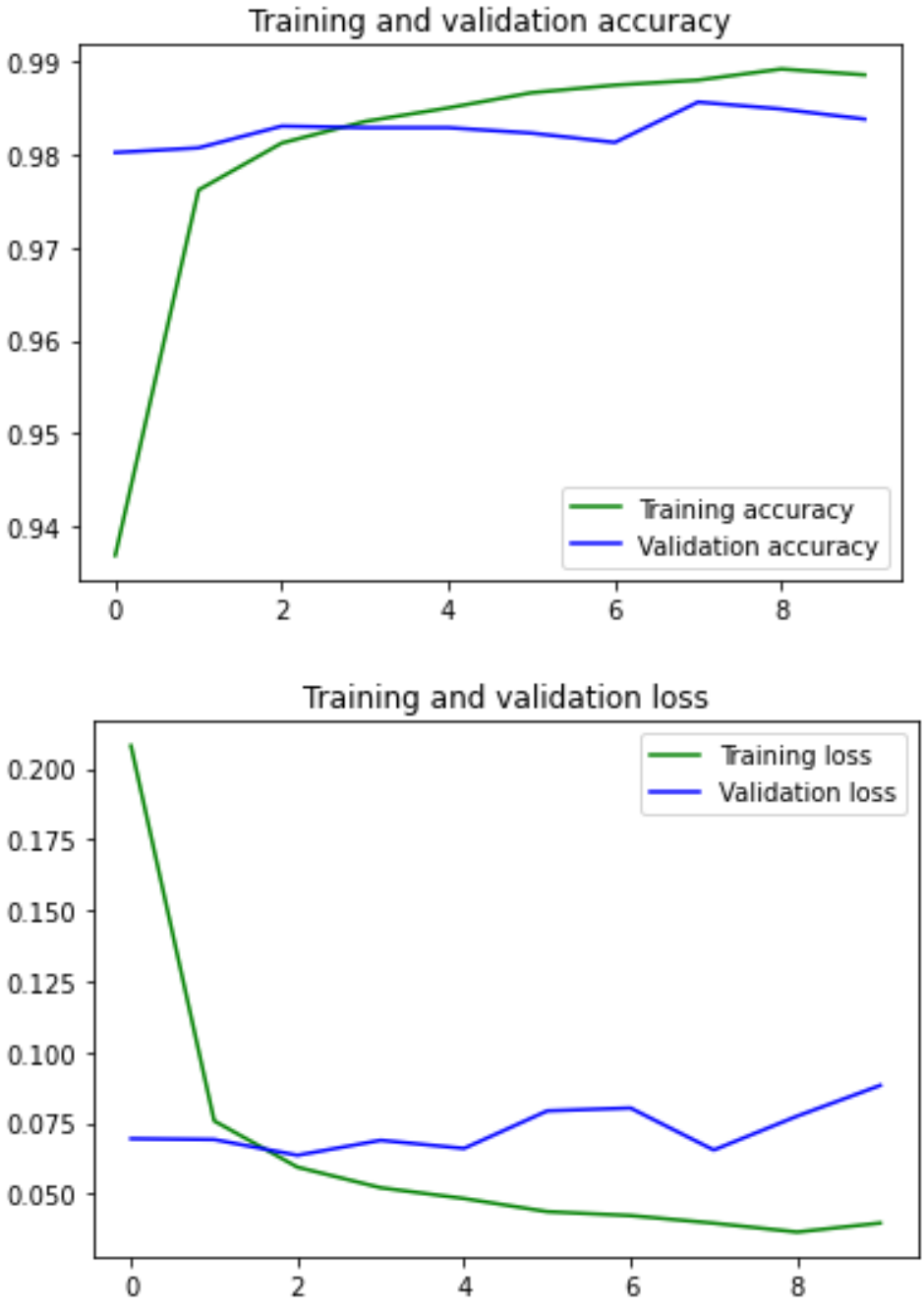
Rysunek 7: Kernel wielkości 2x2



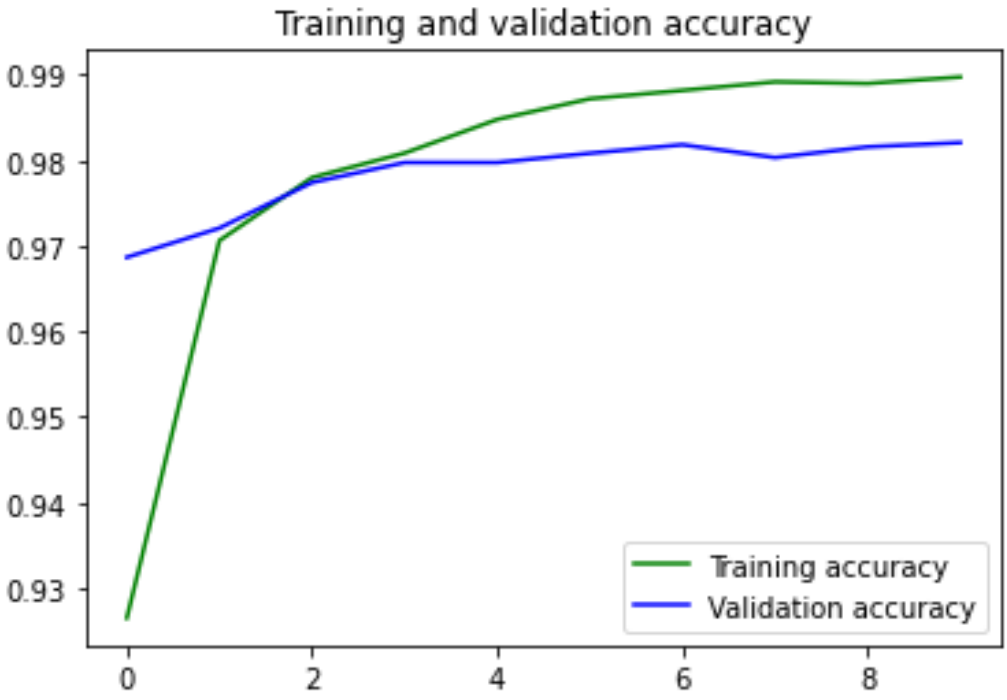
Rysunek 8: Kernel wielkości 3x3

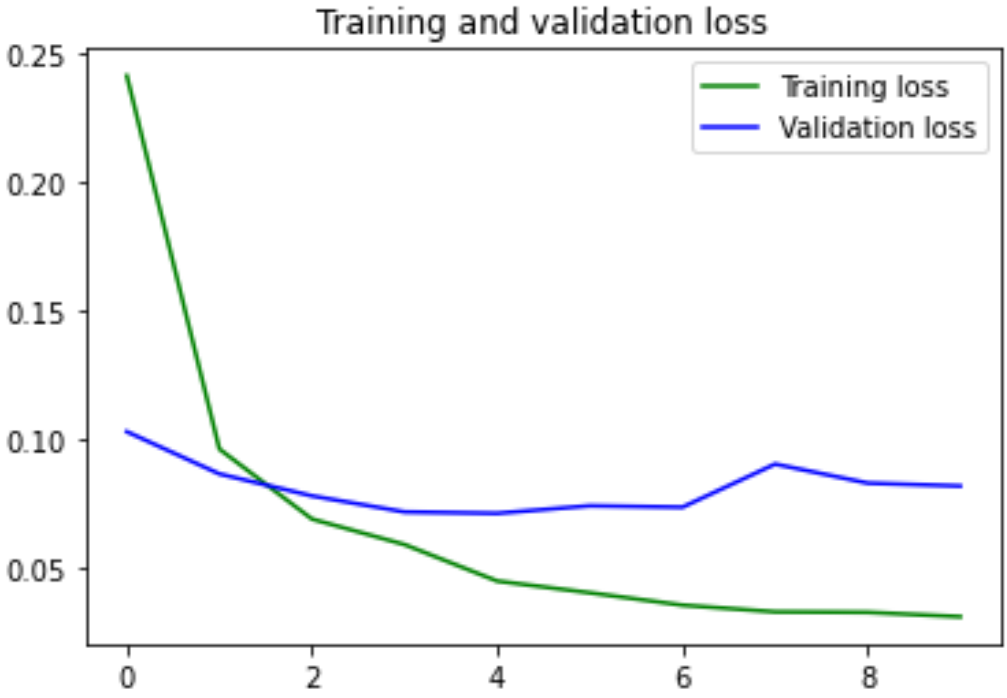


Rysunek 9: Kernel wielkości 5x5



Rysunek 10: Kernel wielkości 7x7





Rysunek 11: Kernel wielkości 11x11

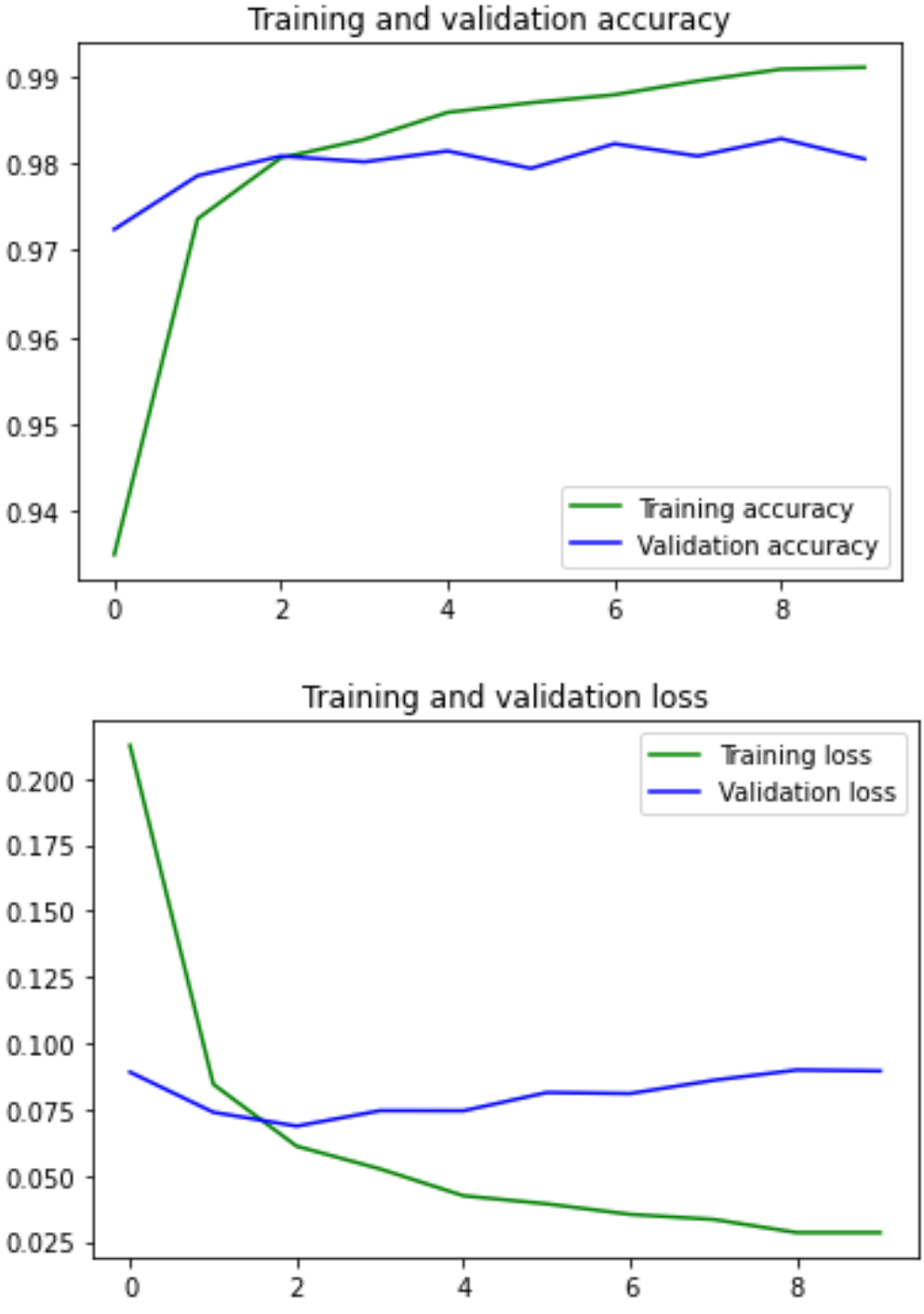
6 Test wielkości mapy cech

Zbadano jak wielkość mapy cech wpływa na skuteczność i prędkość nauki sieci. Przedstawione w tabelce wyniki odnoszą się do wyników sieci na zbiorze testowym. Wykresy odnoszą się do wyników sieci na zbiorze treningowym i walidacyjnym.

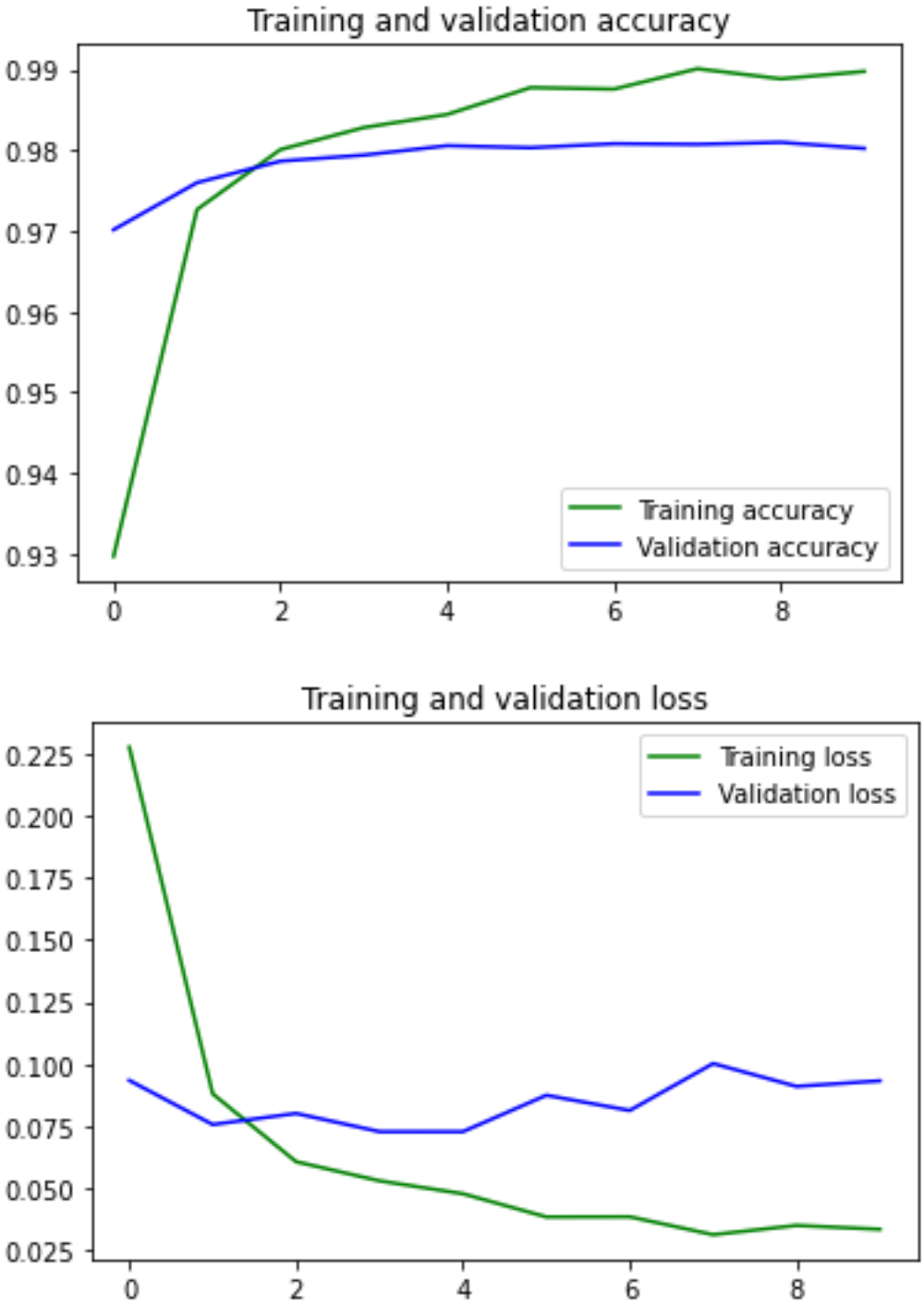
Wielkość mapy cech	Czas trenowania [s]	Osiągnięcie progu [epoka]	Błąd	Skuteczność
32	101	3	0.0878	0.9835
64	189	3	0.0901	0.981
128	343	3	0.0943	0.9793
256	679	4	0.0878	0.9822

Tablica 6: Zastosowanie różnej wielkości mapy cech

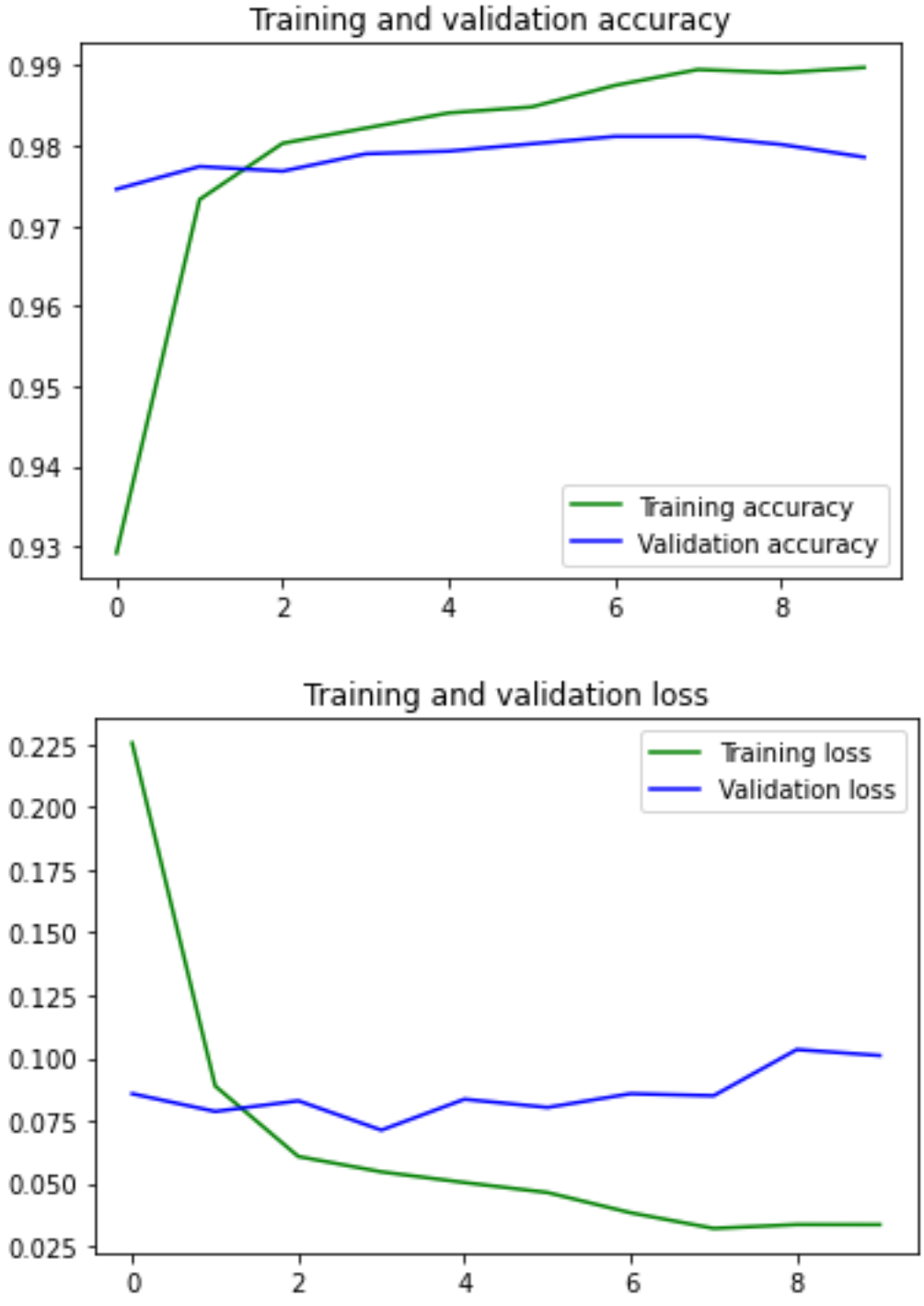
Wielkość mapy cech mówi o tym, jak wiele charakterystycznych cech sieć będzie rozpoznawać na obrazie. Zbyt duża ilość może doprowadzić do przeuczenia sieci, co widoczne jest na wykresach - błąd zbioru walidacyjnego sukcesywnie rośnie. Im więcej map cech tym więcej obliczeń do wykonania, zatem czas wykonania rośnie.



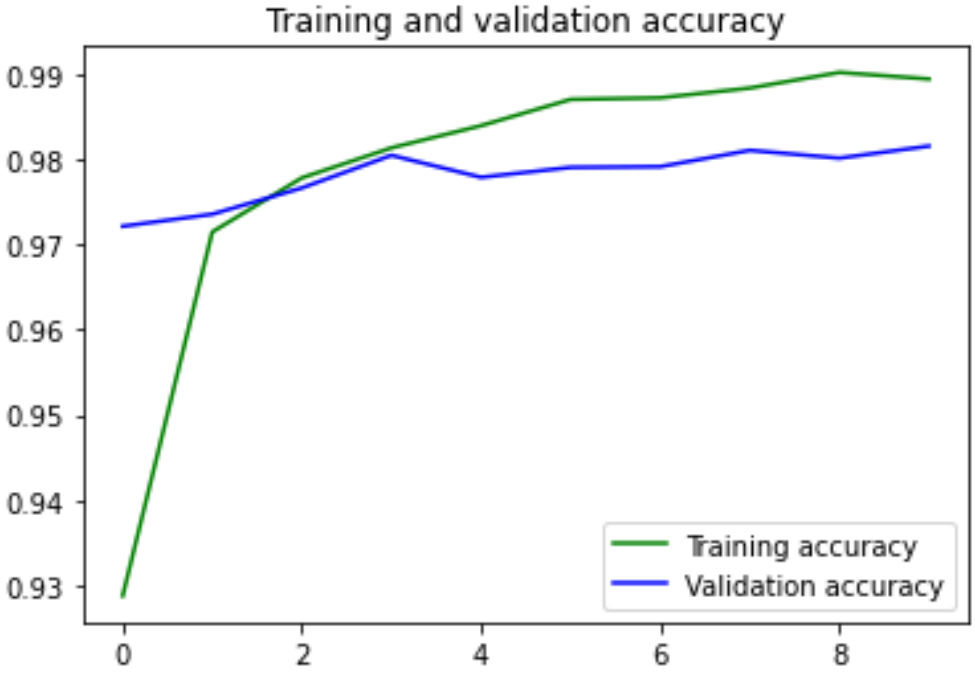
Rysunek 12: Warstwa konwolucyjna z mapą cech o wielkości 32

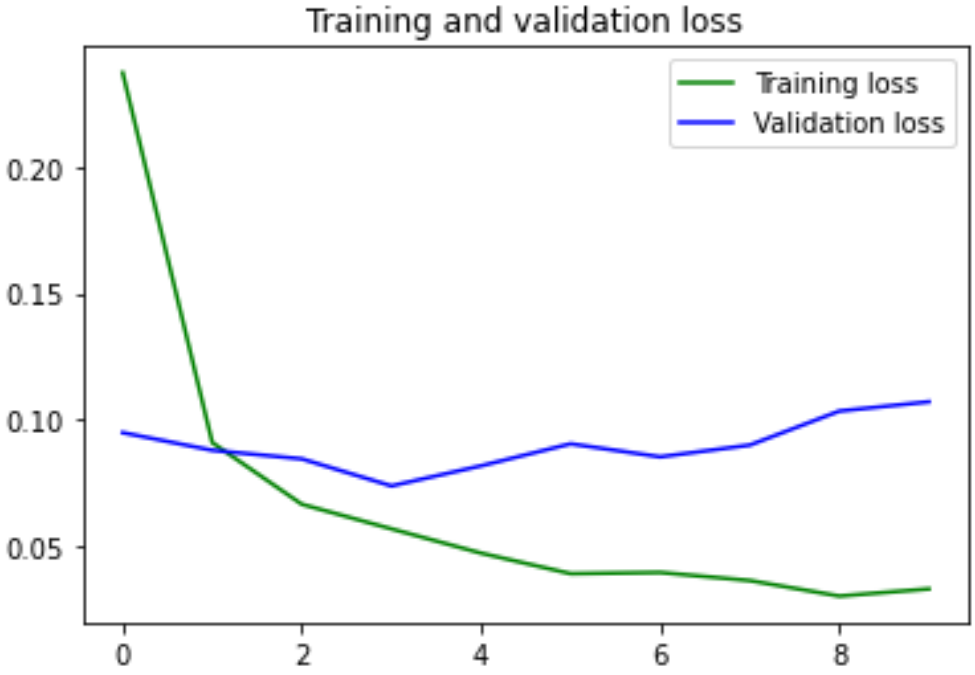


Rysunek 13: Warstwa konwolucyjna z mapą cech o wielkości 64



Rysunek 14: Warstwa konwolucyjna z mapą cech o wielkości 128





Rysunek 15: Warstwa konwolucyjna z mapą cech o wielkości 256

7 Testy poolingu

Zbadano jak wielkość i metoda poolingu wpływa na prędkość nauki i skuteczność sieci.

7.1 Average pooling

Wielkość poolingu	Czas trenowania [s]	Osiągnięcie progu [epoka]	Błąd	Skuteczność
2x2	86	4	0.0776	0.9825
3x3	77	3	0.0682	0.981
5x5	75	7	0.0530	0.9836
10x10	70	-	0.1010	0.9623

Tablica 7: Zastosowanie różnej wielkości mapy cech

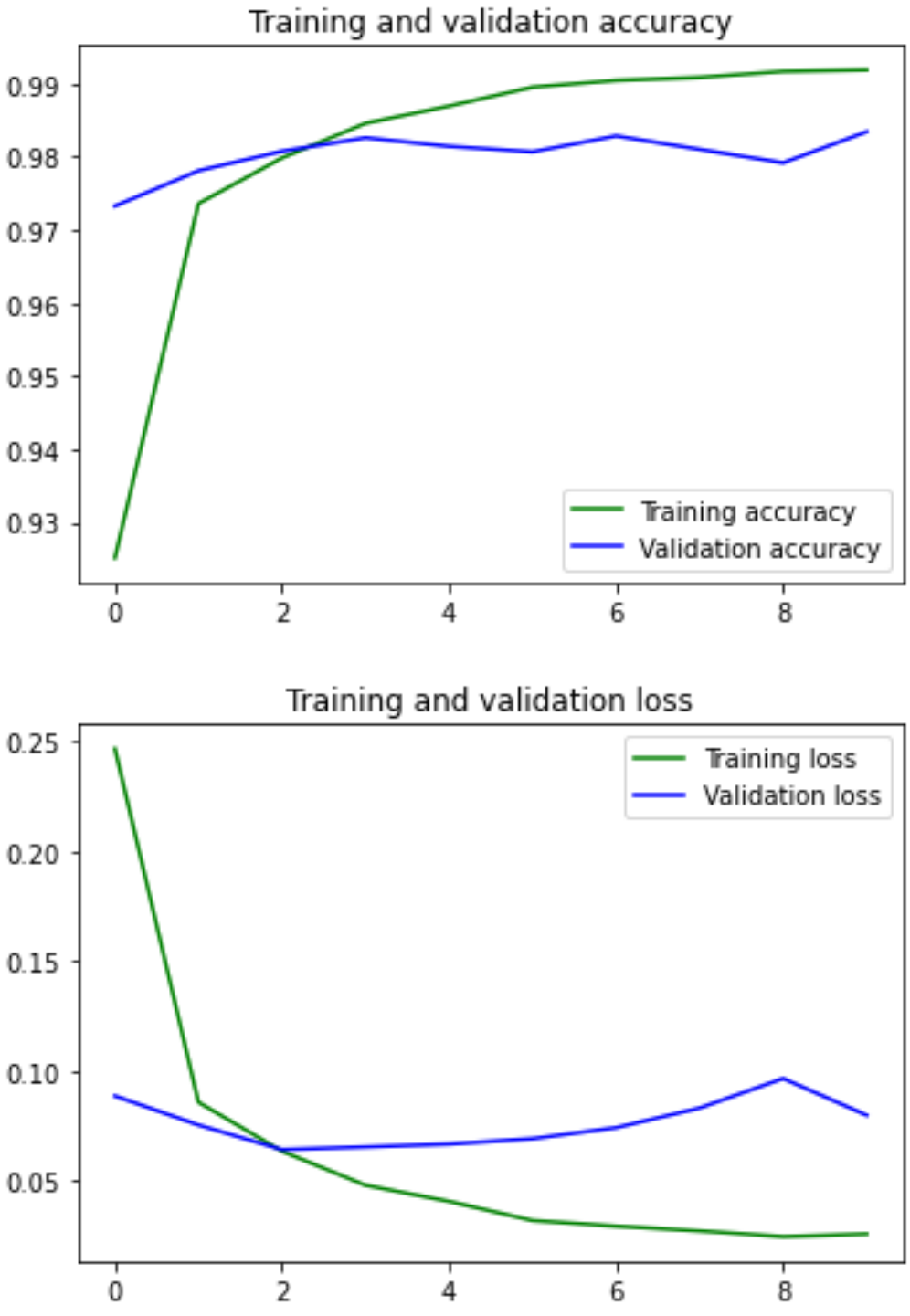
7.2 Max pooling

Wielkość poolingu	Czas trenowania [s]	Osiągnięcie progu [epoka]	Błąd	Skuteczność
2x2	87	4	0.0965	0.9796
3x3	76	2	0.0727	0.9803
5x5	70	4	0.0646	0.9823
10x10	66	-	0.1155	0.9639

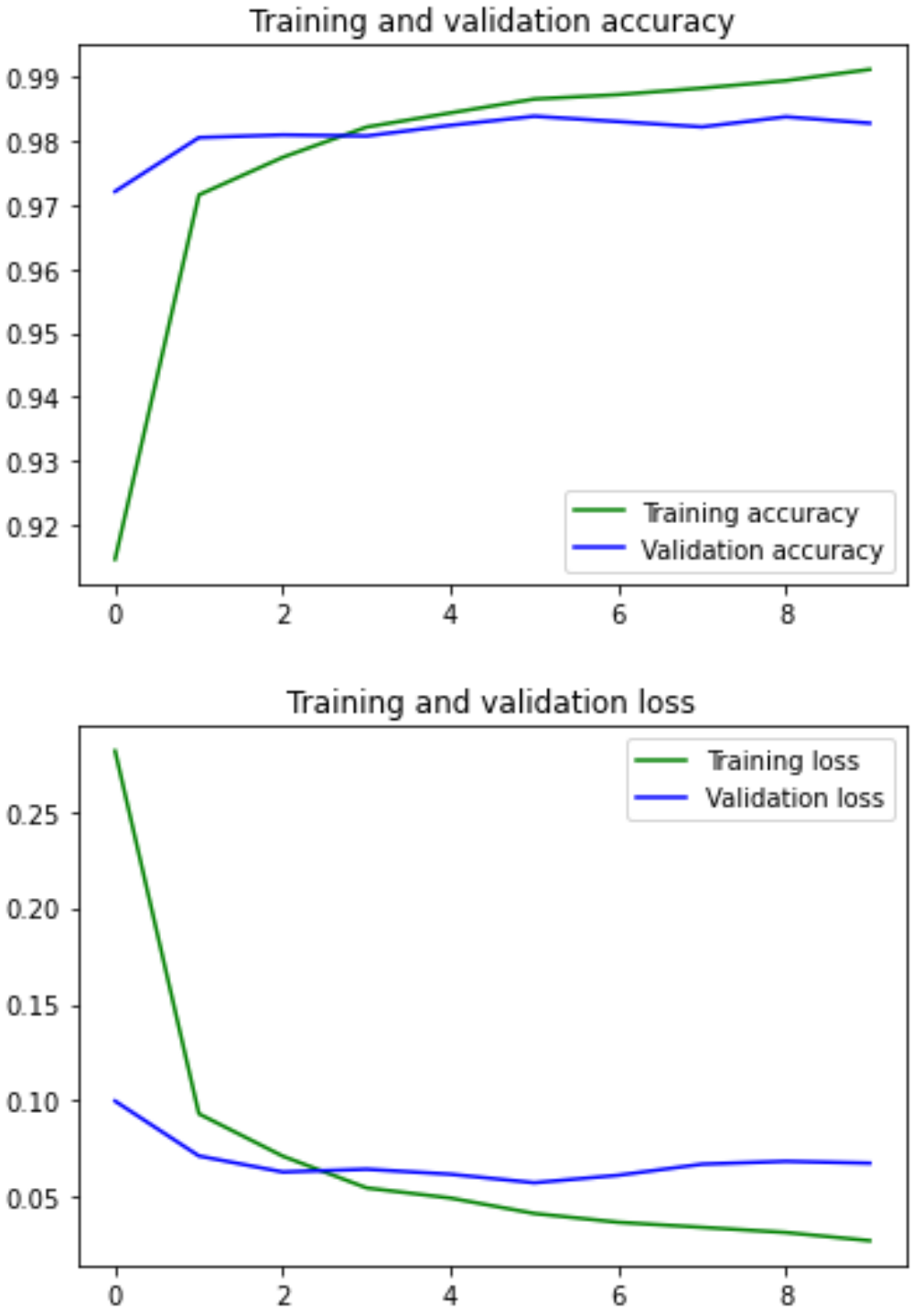
Tablica 8: Zastosowanie różnej wielkości mapy cech

Widać że wraz ze wzrostem wielkości poolingu rośnie nieznacznie prędkość wykonania kodu, jednakże zauważalny jest duży spadek prędkości nauki sieci.

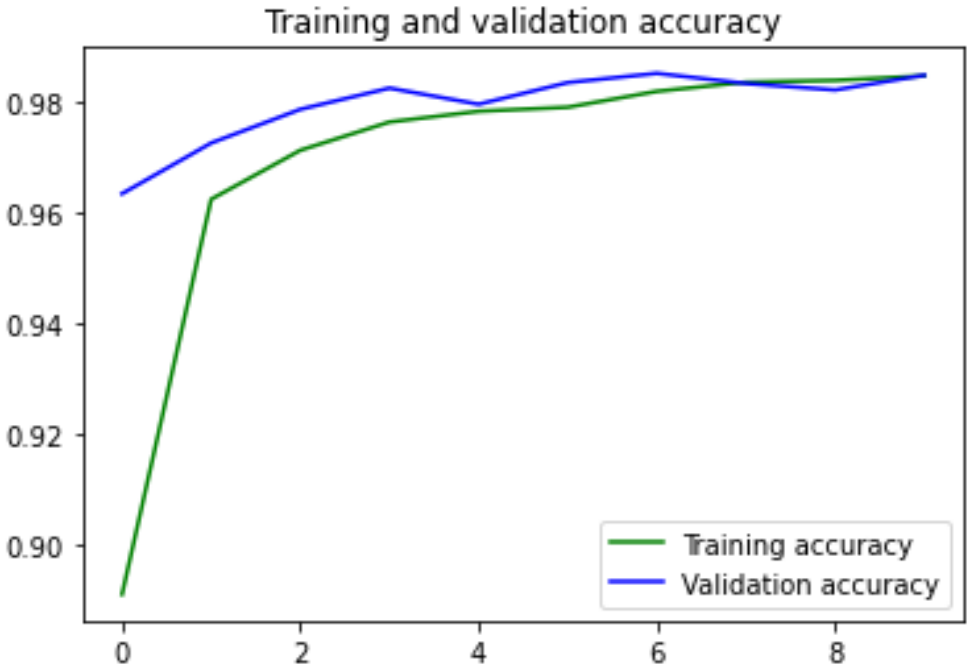
Różnica pomiędy Average poolingiem i Max poolingiem nie jest zauważalna. Z reguły stosowany jest Max pooling.

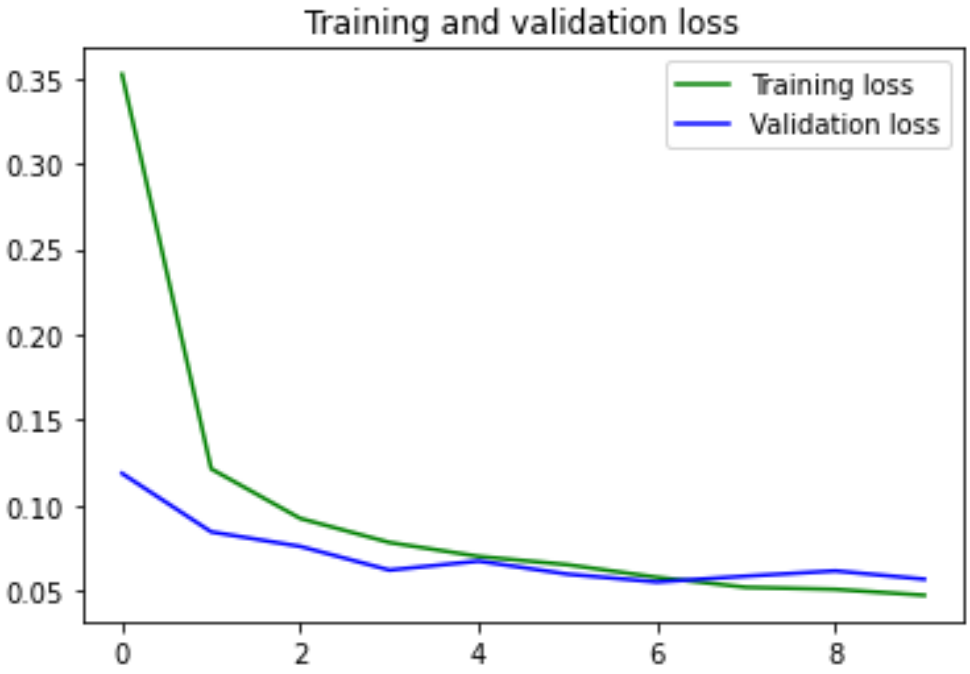


Rysunek 16: Average pooling o wielkości 2x2

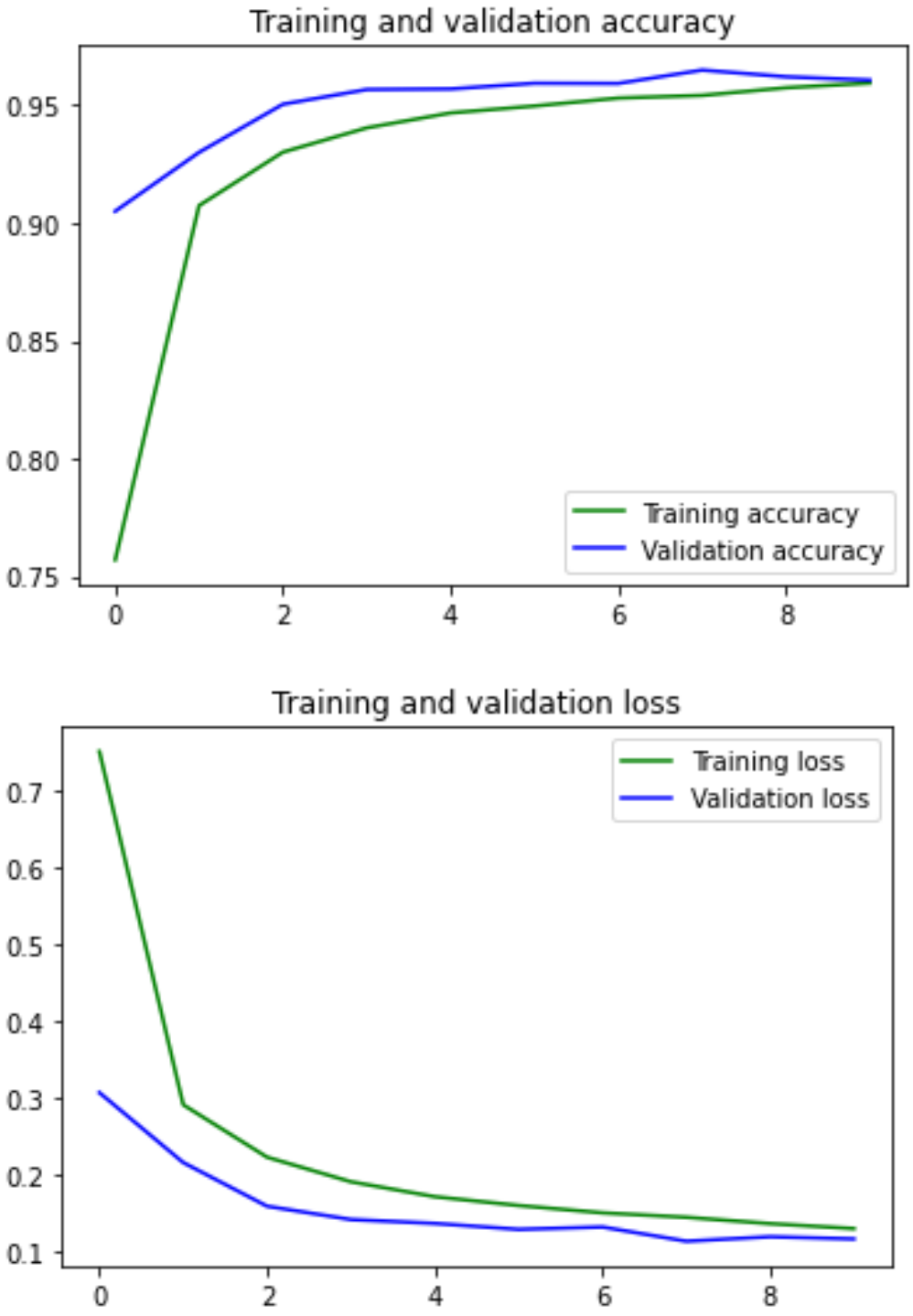


Rysunek 17: Average pooling o wielkości 3x3

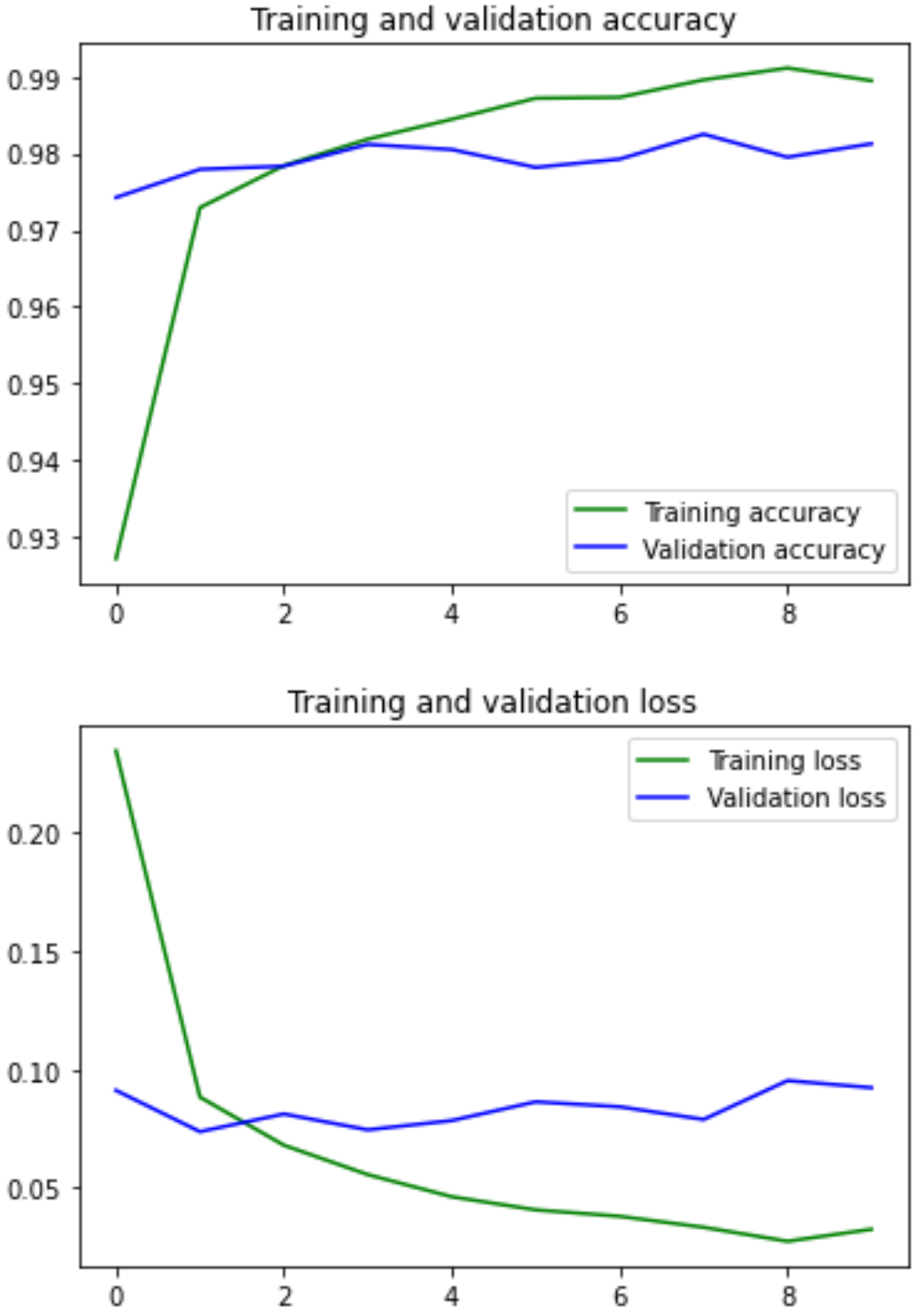




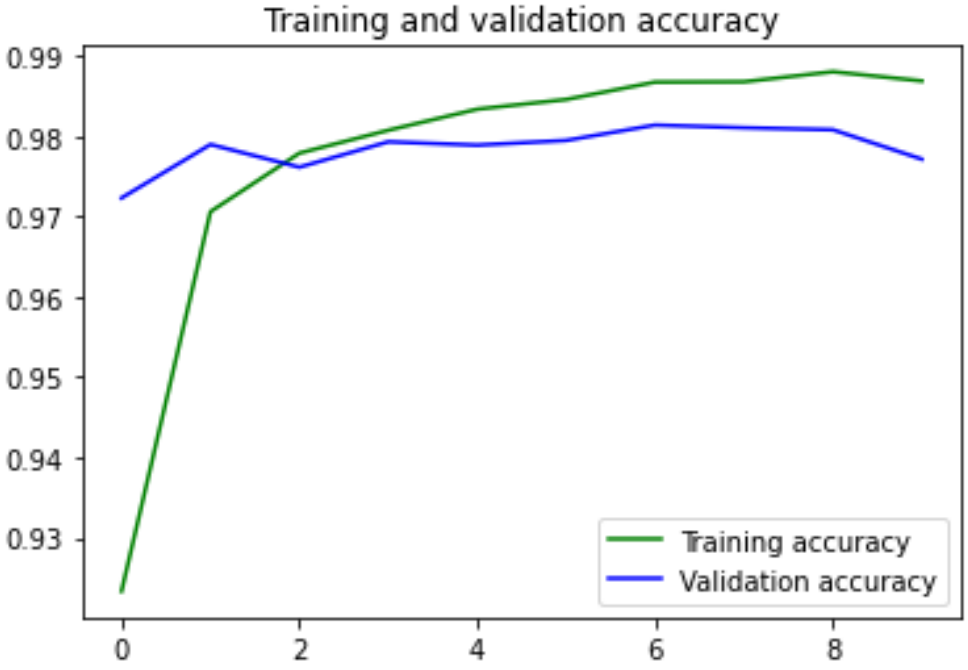
Rysunek 18: Average pooling o wielkości 5x5

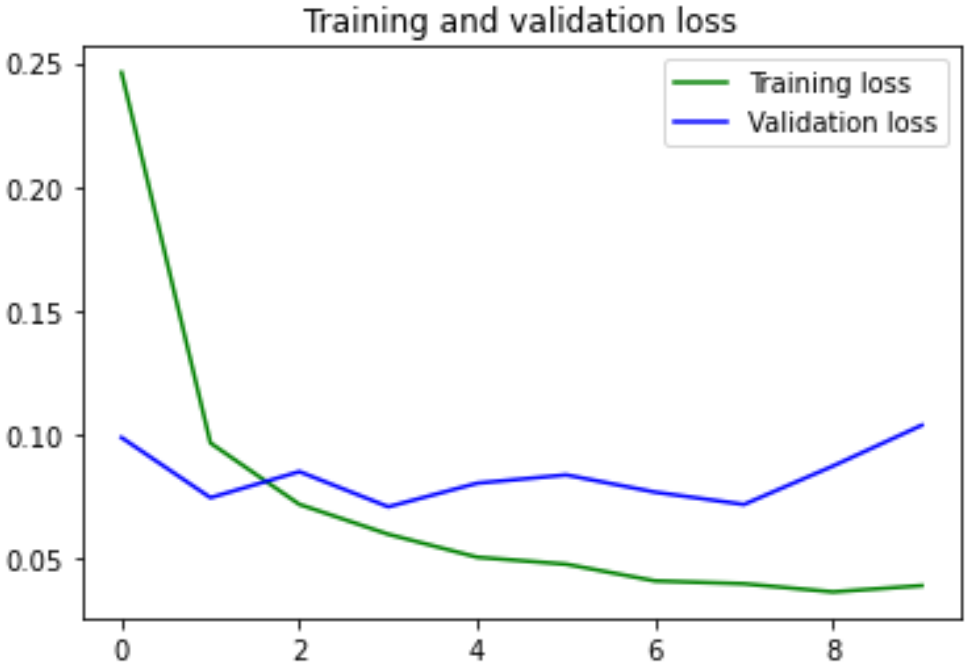


Rysunek 19: Average pooling o wielkości 10x10

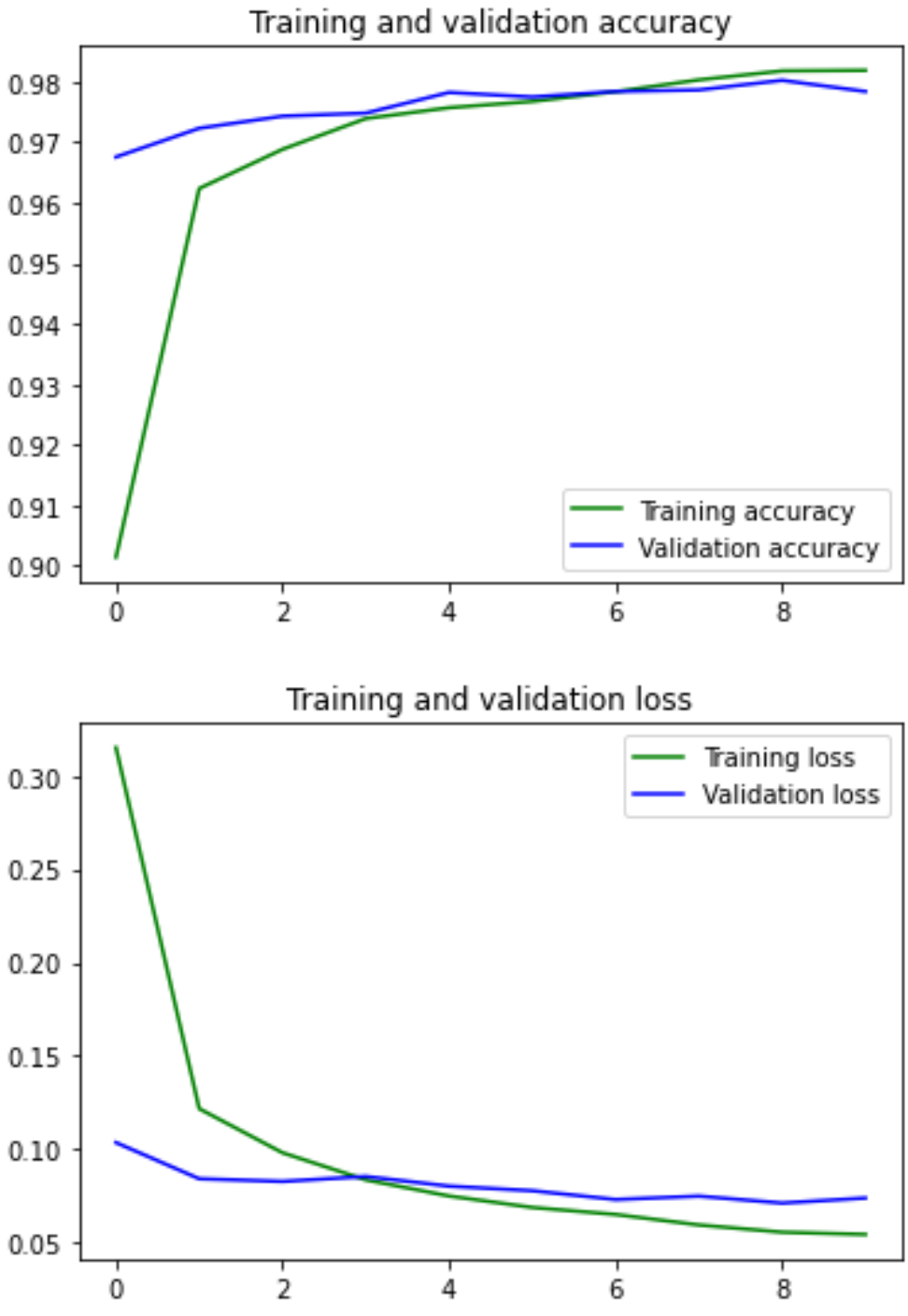


Rysunek 20: Max pooling o wielkości 2x2

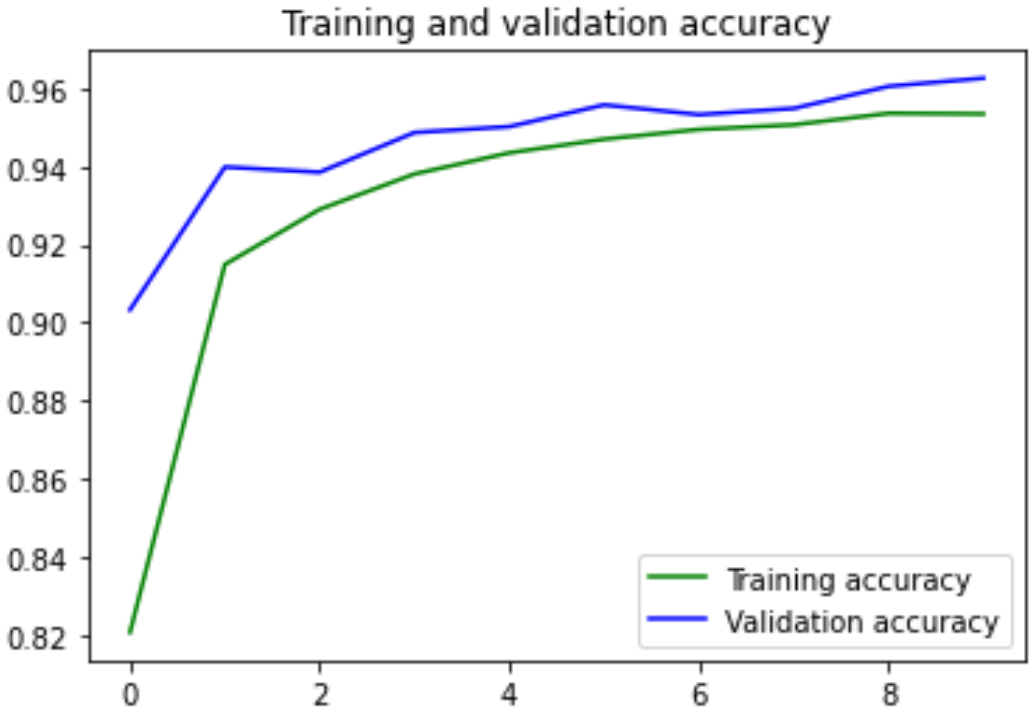


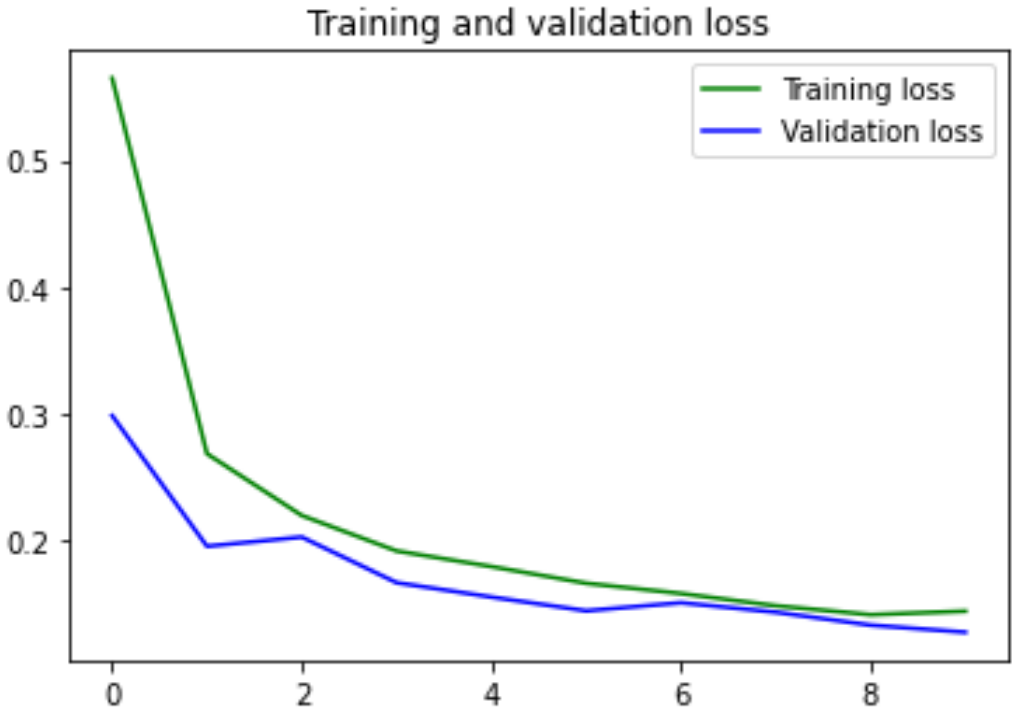


Rysunek 21: Max pooling o wielkości 3x3



Rysunek 22: Max pooling o wielkości 5x5





Rysunek 23: Max pooling o wielkości 10x10

8 Podsumowanie

Warstwy konwolucyjne mogą przycznić się do znacznego usprawnienia działania sieci. Eksperymenty wykazały, że sieć konwolucyjna radzi sobie lepiej ze zbiorem MNIST niż robiła to sieć bez warstw konwolucyjnych. Do tak prostego problemu wykorzystałbym jednak zwykłą sieć bez warstw konwolucyjnych - ze względu na znacznie większą prędkość obliczeń bez dużej straty na skuteczności sieci. Sieć konwolucyjna znacznie lepiej poradzi sobie przy bardziej złożonych problemach jak rozpoznawnie obiektów (także w kolorze), rozpoznawanie twarzy, itp.