Raport 5

Krzysztof Maciejewski 260449

1. Definicja sieci

```
class FashionNN(nn.Module):
    def __init__(self, input_size, output_size, hidden_layers,
hidden_size):
        super(FashionNN, self).__init__()
        self.input_size = input_size
        self.output_size = output_size
        self.hidden_layers = hidden_layers
        self.hidden_size = hidden_size

        self.layers = nn.ModuleList([nn.Linear(input_size,
hidden_size)])

        for _ in range(hidden_layers):
            self.layers.extend([nn.Linear(hidden_size, hidden_size)])

        self.layers.append(nn.Linear(hidden_size, output_size))

    def forward(self, x):
        #pixele do jednowymiarowej macierzy
        x = x.view(-1, self.input_size)
        for layer in self.layers:
            x = torch.relu(layer(x))
        return x
```

Sieć definiuje jako model posiadający dowolną ilość warstw ukrytych. Do zdefiniowania warstw używam obiektu ModelList i Linear. W kroku forward wykorzystuje funkcję view do spłaszczającą piksele obrazów do jednowymiarowej macierzy.

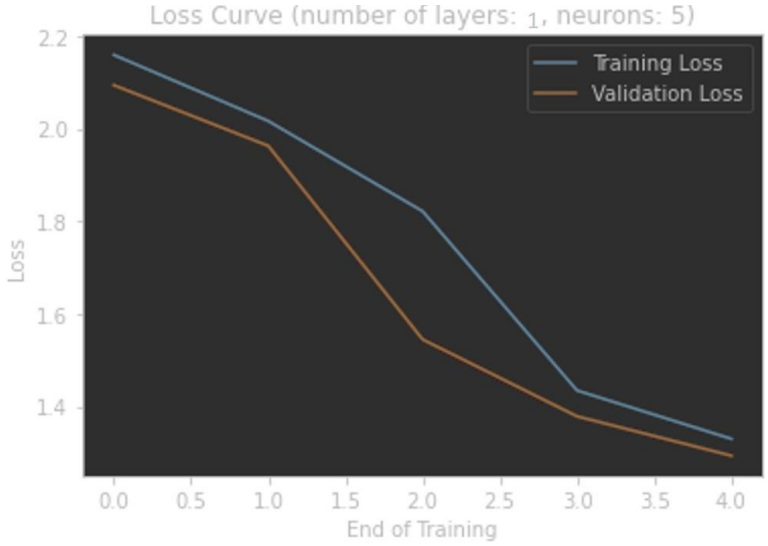
2. Porównanie wyników i krzywych uczenia dla jedno i dwuwarstwowej sieci w zależności od:

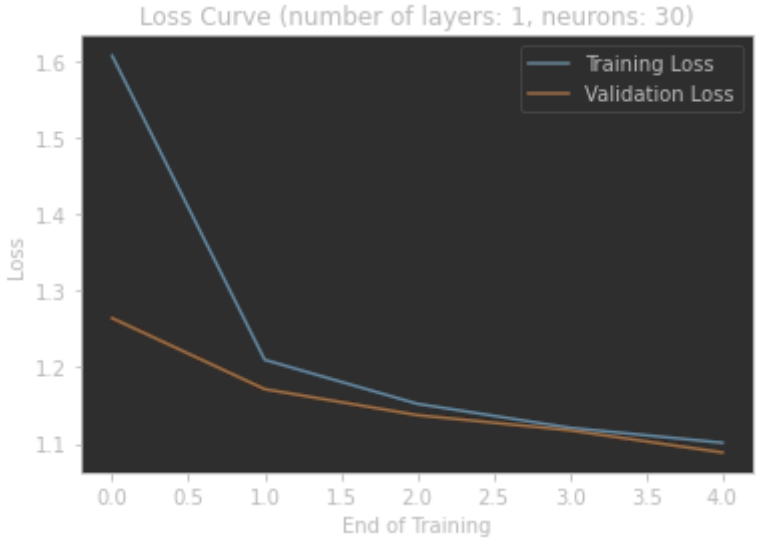
• Liczby neuronów w warstwie ukrytej

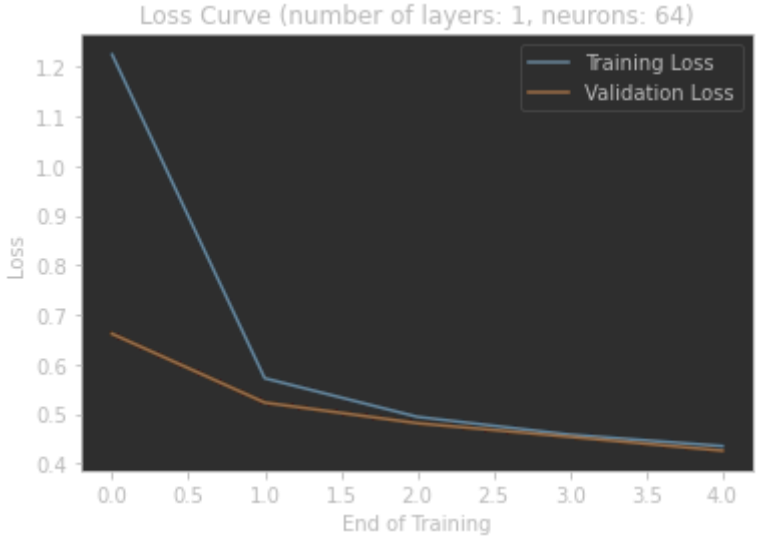
Accuracy:

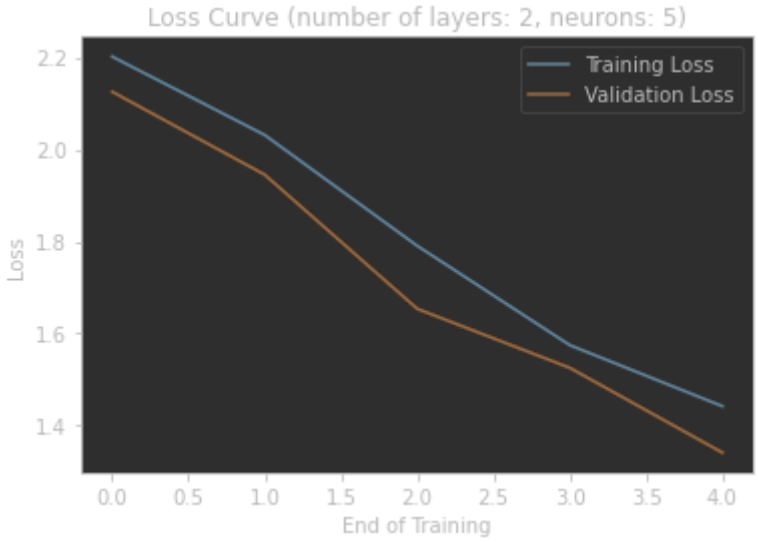
Liczba warstw	Neuronów = 5	Neuronów = 30	Neuronów = 64
1	0.5565	0.5764	0.8492
2	0.5615	0.7853	0.71755

Trochę lepiej poradziły sobie modele mające dwie warstwy ukryte. Więcej warstw pozwoliło im na zidentyfikowanie bardziej złożonych relacji w danych. Najlepiej ze wszystkich modeli poradził sobie ten z jedną warstwą ukrytą i 64 neuronami, były to optymalne parametry pozwalające na dobre zrozumienie relacji w danych i ich szybkie wytrenowanie.

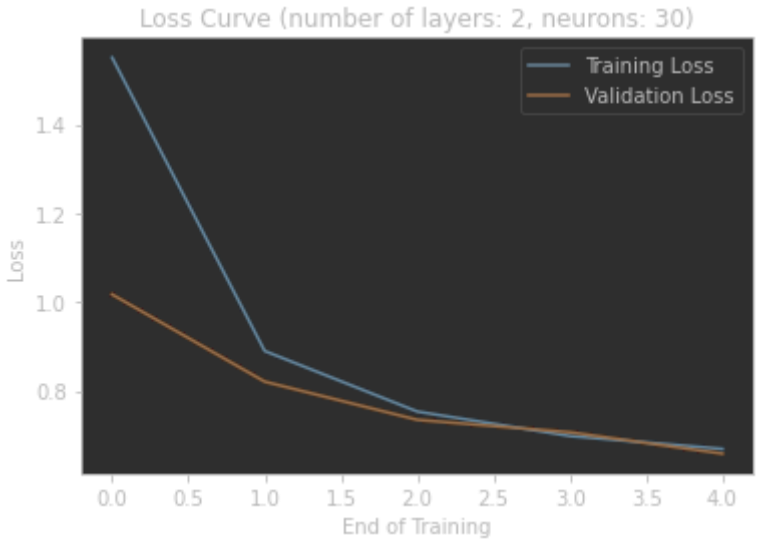

Loss Curve (number of layers: 1, neurons: 5)
Training Loss
Validation Loss
Loss
End of Training

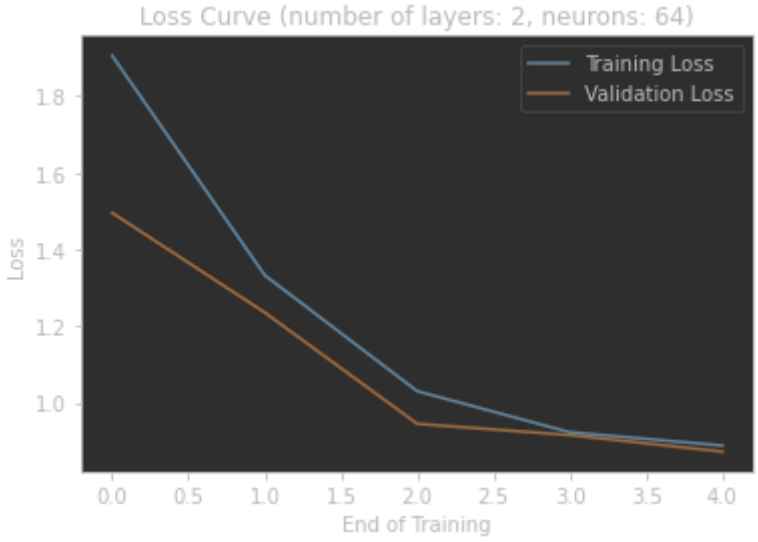

Loss Curve (number of layers: 1, neurons: 30)
Training Loss
Validation Loss
Loss
End of Training


Loss Curve (number of layers: 1, neurons: 64)
Training Loss
Validation Loss
Loss
End of Training



Widać, że model z tylko 5 neuronami w warstwach ukrytych potrzebował więcej epok niż 5, by dostosować się do danych.

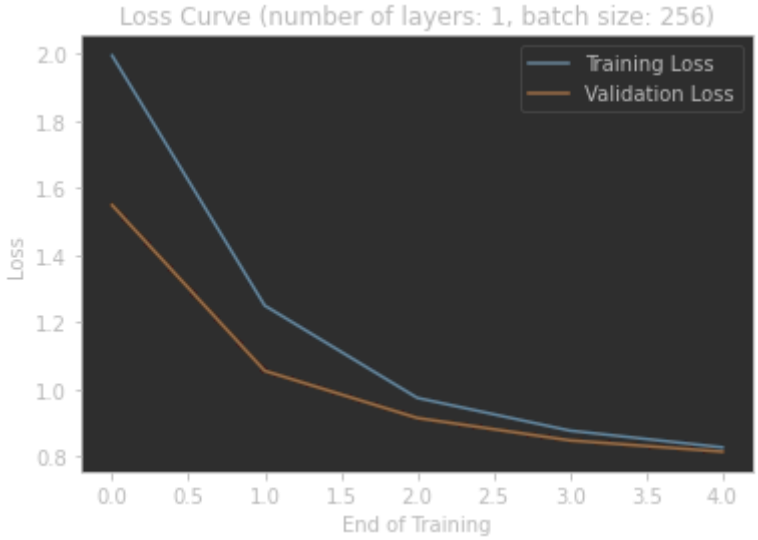


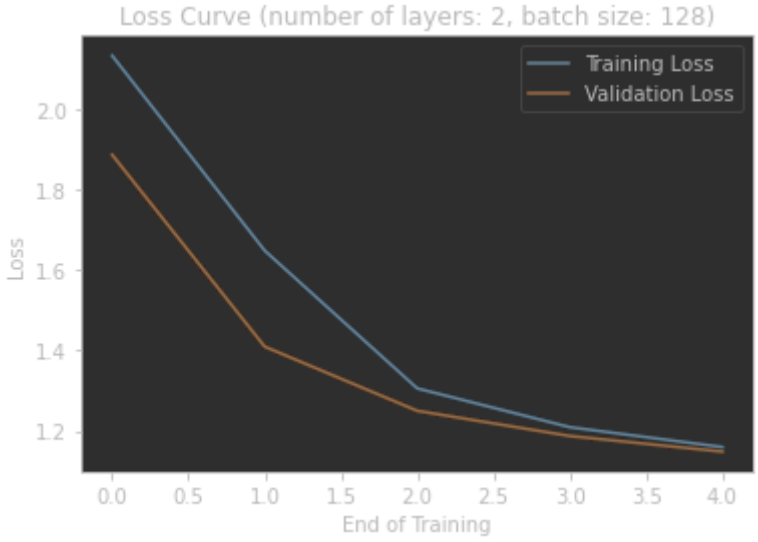


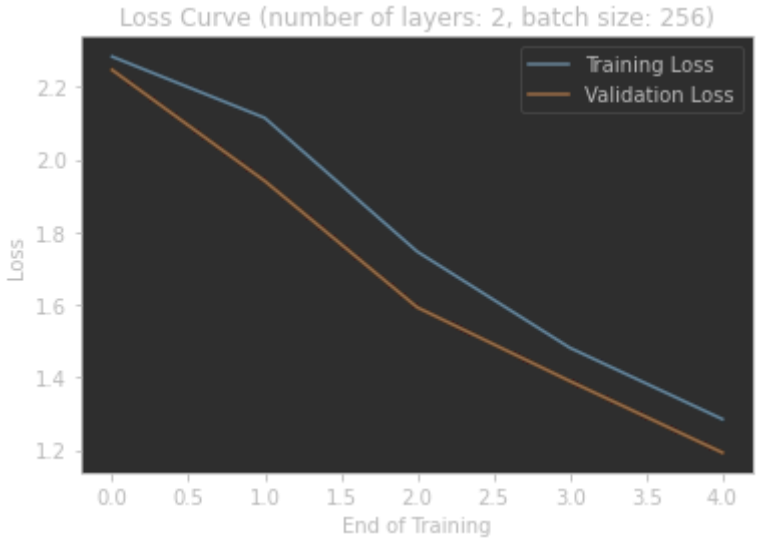
• Rozmiaru batcha

W przypadku badania wpływu rozmiaru batcha trudno było wyciągnąć wartościowe wnioski ponieważ nie zauważyłem regularności.

Liczba warstw	Batch = 64	Batch = 128	Batch = 256
1	0.7311	0.6104	0.7139
2	0.71755	0.6399	0.6579



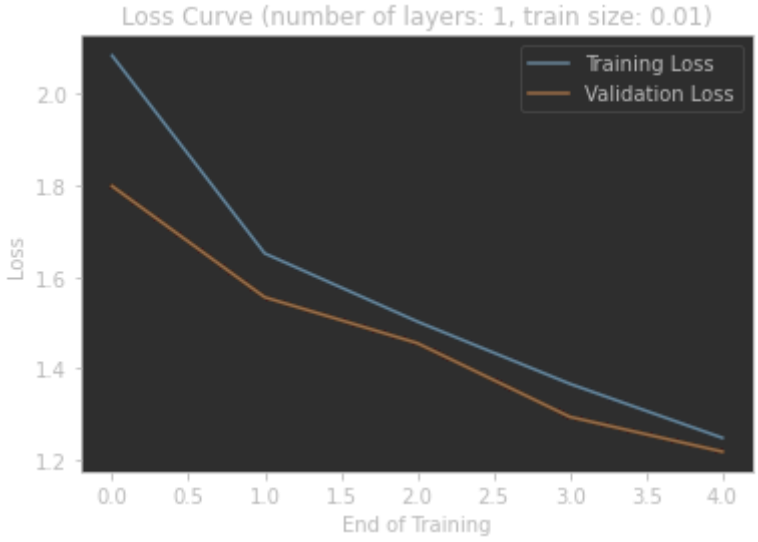


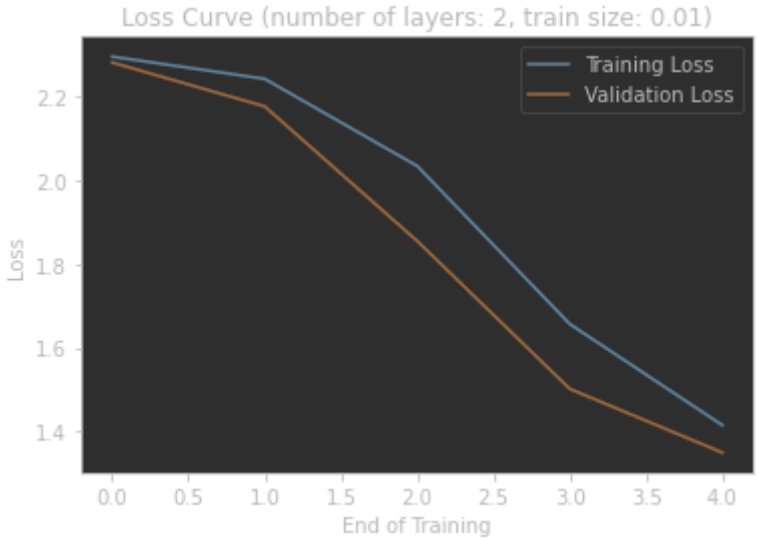


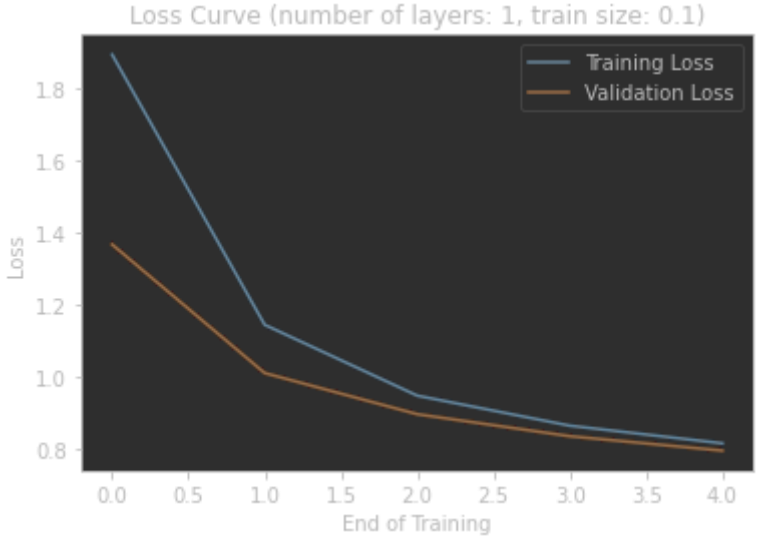
• Liczby przykładów uczących

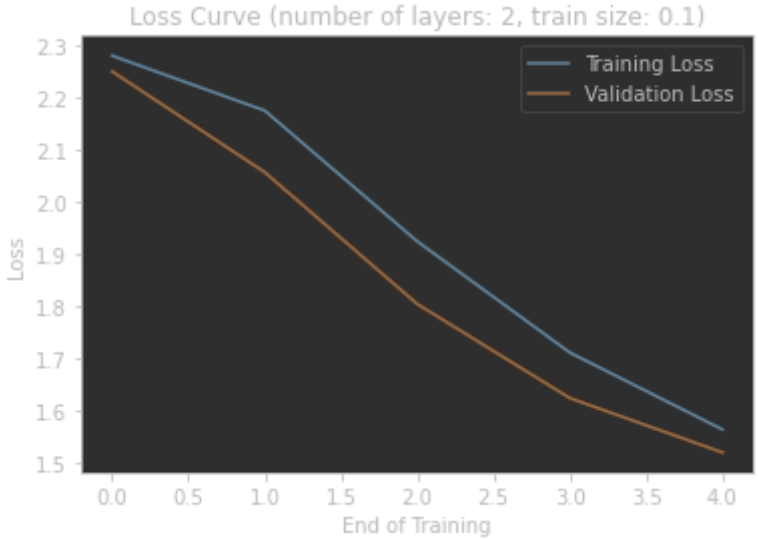
Zwiększenie liczby przykładów uczących zgodnie z oczekiwaniami pozwoliło sieci na szybsze bardziej efektywne uczenie. Zaskakujący wynik był dla sieci od 2 warstwach. Prawdopodobnie spowodowała go mała liczba epok.

Liczba warstw	Rozmiar = 1%	Rozmiar = 10%
1	0.5831	0.7619
2	0.5239	0.4759




Loss Curve (number of layers: 2, train size: 0.01)
Training Loss
Validation Loss
Loss
2.2
2.0
1.8
1.6
1.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
End of Training

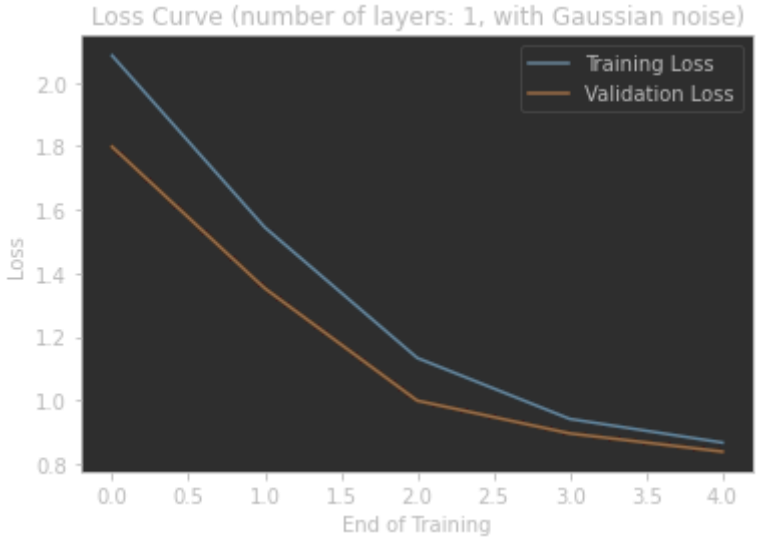

Loss Curve (number of layers: 1, train size: 0.1)
Training Loss
Validation Loss
Loss
1.8
1.6
1.4
1.2
1.0
0.8
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
End of Training

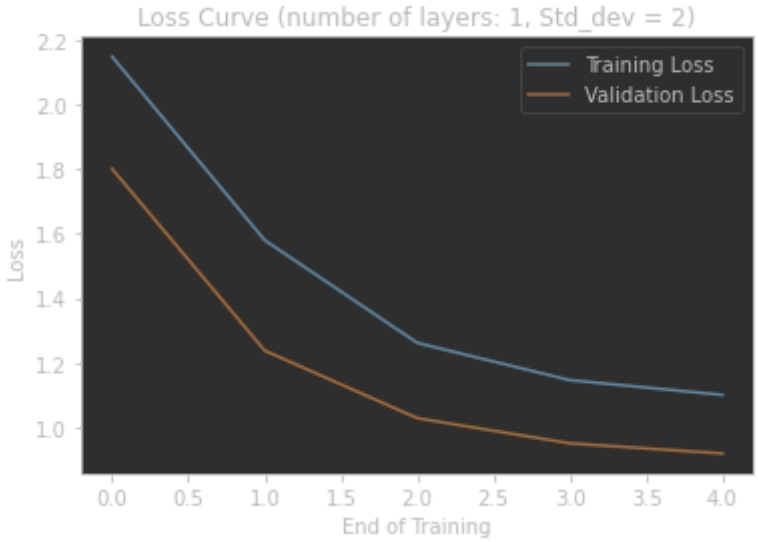

Loss Curve (number of layers: 2, train size: 0.1)
Training Loss
Validation Loss
Loss
2.3
2.2
2.1
2.0
1.9
1.8
1.7
1.6
1.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
End of Training

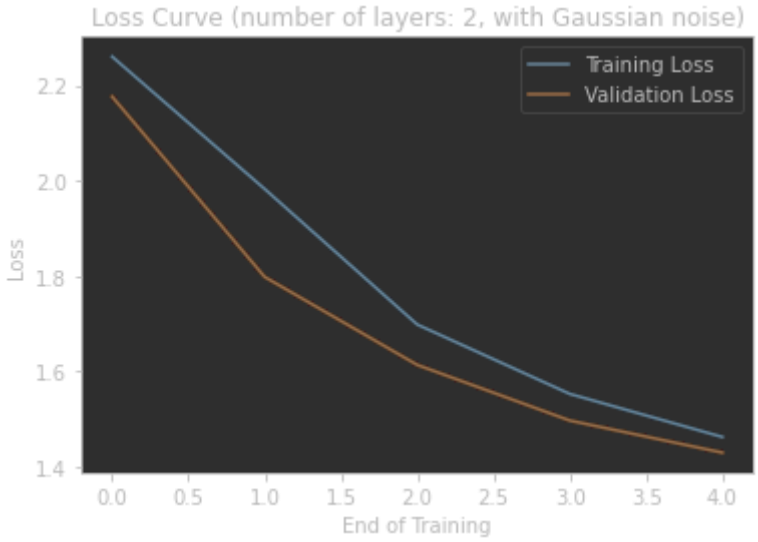
• Zaburzenia danych:

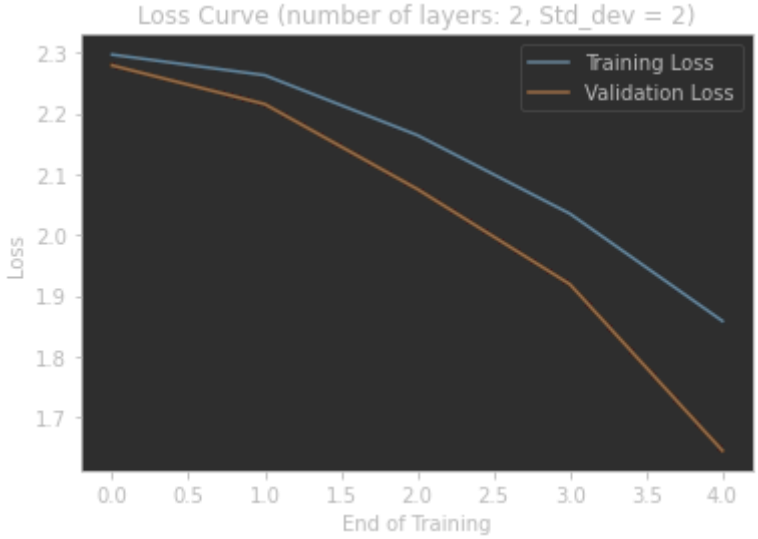
- szum dodany w danych testowych

Liczba warstw	Odchylenie standardowe = 0.1	Odchylenie standardowe = 2
1	0.7181	0.6143
2	0.5177	0.4436



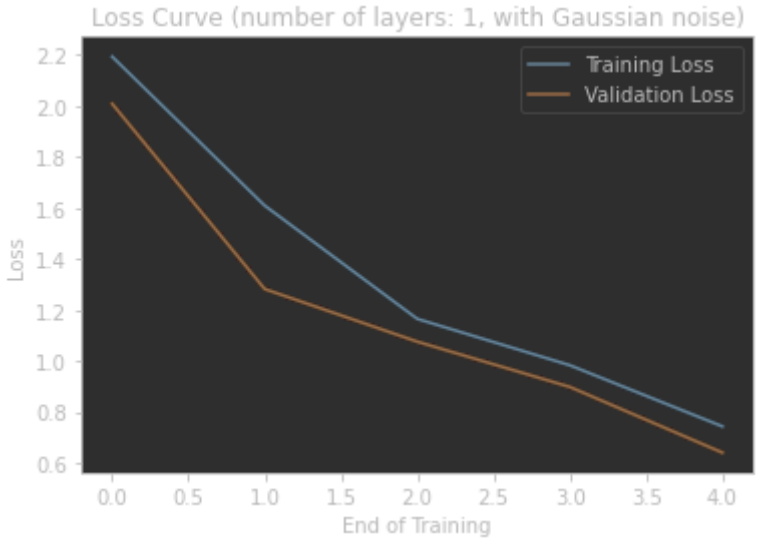


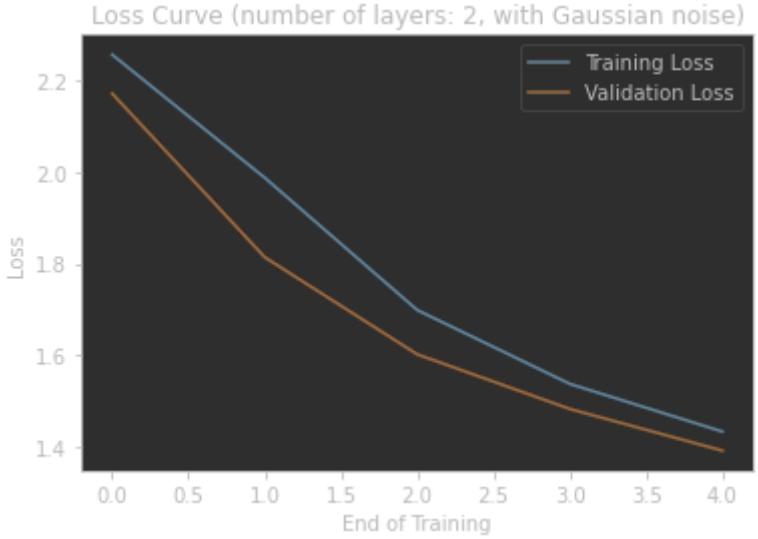


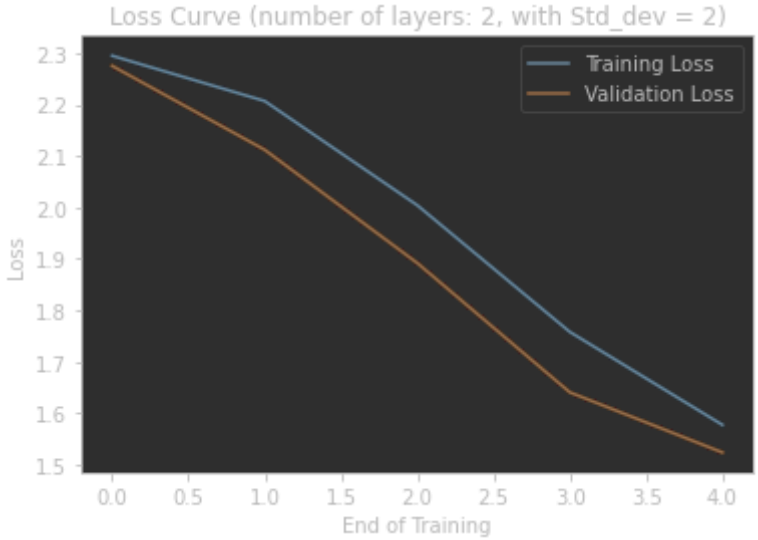


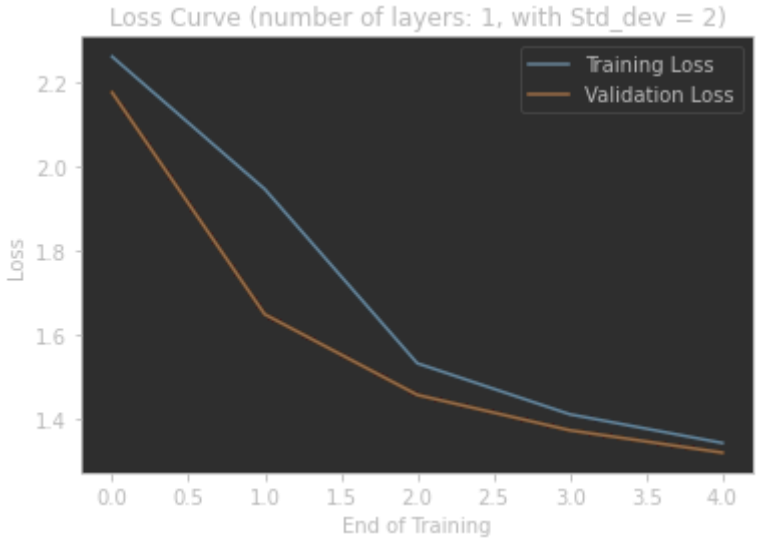
- szum dodany w testowych i treningowych.

Liczba warstw	Odchylenie standardowe = 0.1	Odchylenie standardowe = 2
1	0.7735	0.5917
2	0.5917	0.5429


Loss Curve (number of layers: 1, with Gaussian noise)
Training Loss
Validation Loss
Loss
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
End of Training


Loss Curve (number of layers: 2, with Gaussian noise)
Training Loss
Validation Loss
Loss
2.2
2.0
1.8
1.6
1.4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
End of Training


Loss Curve (number of layers: 2, with Std_dev = 2)
Training Loss
Validation Loss
Loss
2.3
2.2
2.1
2.0
1.9
1.8
1.7
1.6
1.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
End of Training



Wnioski

W wielu przypadkach modele z dwoma warstwami poradziły sobie gorzej. Analizując ich wykresy funkcji kosztu, stwierdziłem że jest to prawdopodobnie spowodowane zbyt małą liczbą epok, która nie pozwala na szybkie i efektywne wytrenowanie dużej liczby wag. Więcej warstw oznacza więcej wag pozwalających na wychwycenie bardziej skomplikowanych zależności w danych, jednakże trzeba je dłużej trenować.

Szum dodany w danych testowych i treningowych poradził sobie minimalnie lepiej od przypadku z szumem tylko w danych testowych. Dodanie szumu do obu zestawów danych pomogło w utrzymaniu równowagi pomiędzy nimi.Szum wprowadza różnorodność do danych treningowych, co może pomóc modelowi w lepszym uchwyceniu ogólnych wzorców, a nie tylko dopasowywaniu się do konkretnych przypadków treningowych.