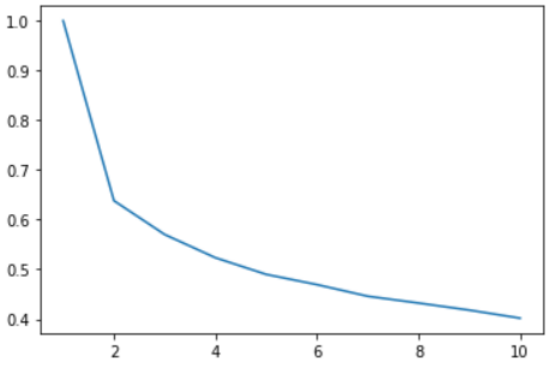
Ekstrakcja Cech

```python
def extractFeatures(self, x):
    x = x.view(-1, 28, 28)
    maxV, maxI = torch.max(x, 1)
    minV, minI = torch.min(x, 1)
    medV1, medI1 = torch.median(x, dim=1)
    medV2, medI2 = torch.median(x, dim=2)
    return torch.concat([maxV, maxI, minV,
```

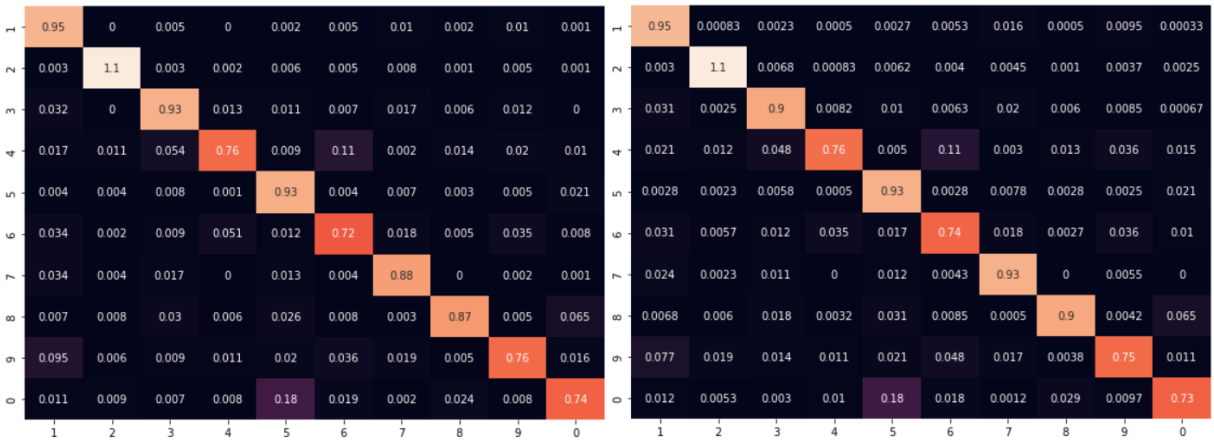
W tym wariancie ekstrahowane są wartości oraz współrzędne maksymalnej, minimalnej wartości oraz mediana. Następnie cechy przechodzą przez trzy warstwy liniowe.

```python
nn.Linear(28 * 8, 84)
nn.Linear(84, 42)
nn.Linear(42, 10)
```

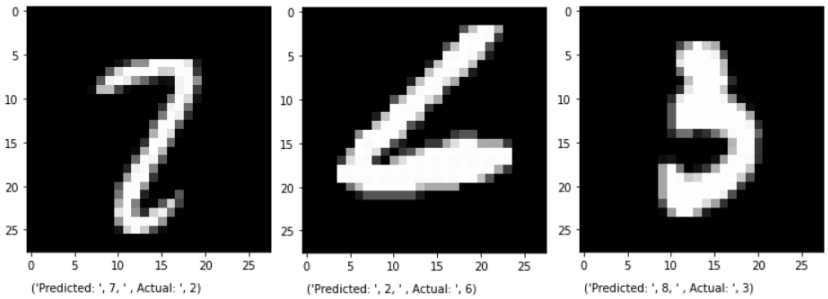
Użytą funkcją celu jest Cross Entropy Loss a optymalizatorem Adam. Nauka odbywała się dla dziesięciu epok.



Czas potrzebny do nauki wynosi ***2m 33,7s*** dla CPU: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz i pamięci RAM 16,00 GB. Skuteczność wyniosła 86,31%.



Macierz pomyłek dla zbioru testowego i treningowego



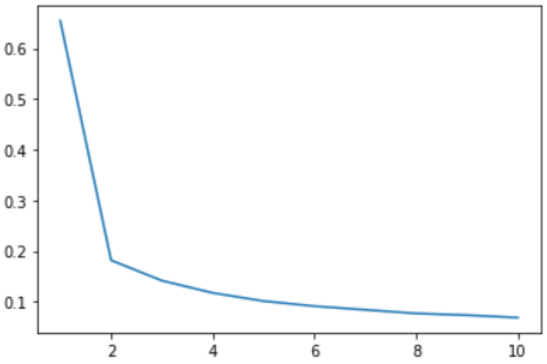
Przykłady cyfr, które zostały niepoprawnie sklasyfikowane.

Algorytm sprawdza się dość dobrze jednak nie radzi sobie dobrze z cyframi takimi jak 6, 4, 9 oraz 0. Najwięcej razy myli zero z piątką oraz czwórkę z szóstką.

Własna Architektura

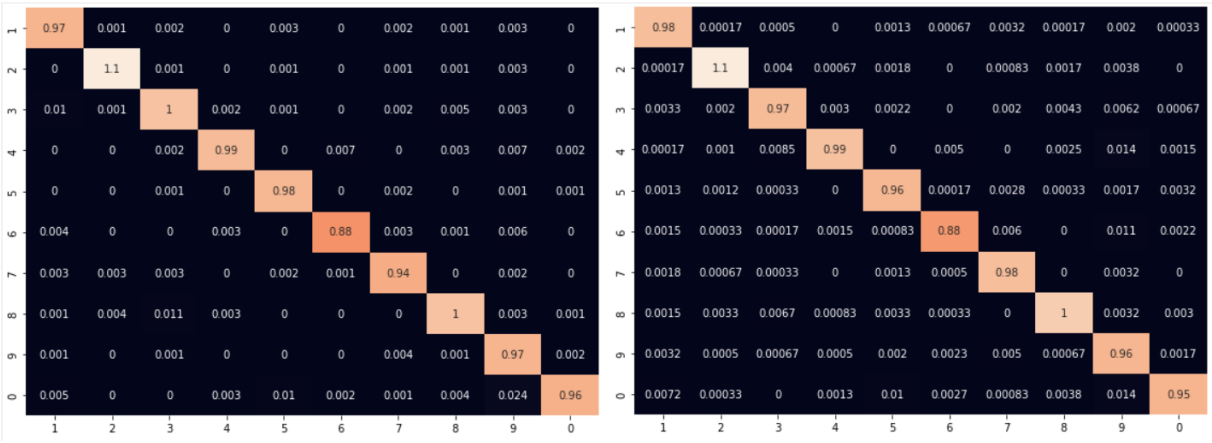
```
model = nn.Sequential(
    nn.Conv2d(in_channels=1,
    nn.ReLU(),
    nn.MaxPool2d(kernel_size
    nn.Conv2d(in_channels=7,
    nn.ReLU(),
    nn.MaxPool2d(kernel_size

    nn.Flatten(),
    nn.Linear(56, 28),
    nn.ReLU(),
    nn.Linear(28, 28),
    nn.ReLU(),
    nn.Linear(28, 10)
)
```

W tym wariancie została użyta własna sieć. Składa się ona z czterech warstw. Dwie z nich to warstwy splotowe po których wykonywana jest funkcja aktywacji ReLU. Po każdej z warstw splotowych wykonywana jest operacja max poolingu. Następnie jest wykonywane spłaszczenie oraz trzy warstwy liniowe.

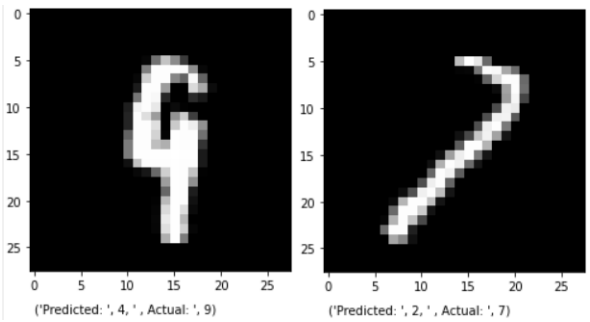


Użytą funkcją celu jest Cross Entropy Loss a optymalizatorem Adam. Nauka odbywała się dla dziesięciu epok.

Czas potrzebny do nauki wynosi ***4m 35,3s*** dla CPU: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz i pamięci RAM 16,00 GB. Skuteczność wyniosła 98,19%.



Macierz pomyłek dla zbioru testowego i treningowego



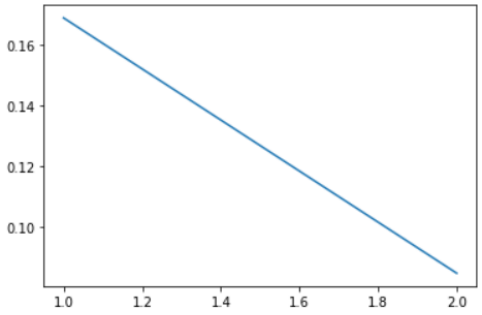
Przykłady cyfr, które zostały niepoprawnie sklasyfikowane.

Algorytm sprawdza się dość dobrze jednak nie zawsze rozpoznaje cyfry poprawnie a najgorzej radzi sobie z rozpoznawaniem szóstek. Na obu macierzach pomyłek procent rozpoznanych cyfr to 88%.

Transfer Sieci – uczenie części sieci

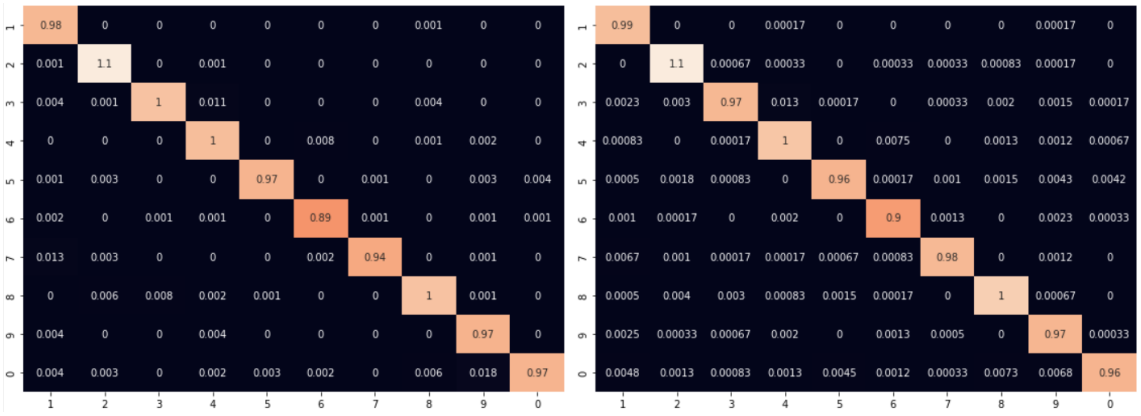
```
classifier = nn.Sequential(
    nn.Dropout(),
    nn.Linear(9216, 4096),
    nn.ReLU(),
    nn.Dropout(),
    nn.Linear(4096, 4096),
    nn.ReLU(),
    nn.Linear(4096, 10),
)
```

W tym wariancie została użyta sieć alexnet oraz własny klasyfikator. Składa się on z czterech warstw. Trzy z nich to warstwy liniowe po których wykonywana jest funkcja aktywacji ReLU. Pierwszą warstwą jest warstrwa dropout. Następnie wykonywane są wcześniej wspomniane warstwy liniowe.

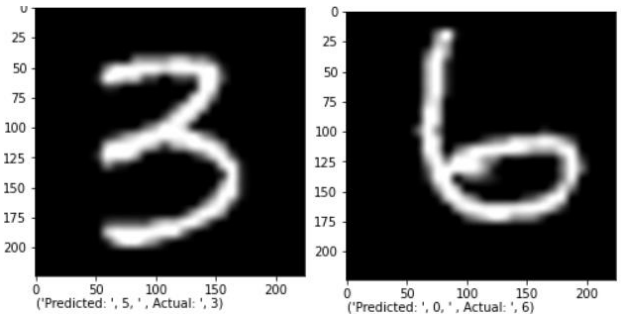


Użytą funkcją celu jest Cross Entropy Loss a optymalizatorem SDG. Nauka odbywała się dla dwóch epok.

Czas potrzebny do nauki wynosi ***301m 13,2s*** dla CPU: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz i pamięci RAM 16,00 GB. Skuteczność wyniosła 98,64%.



Macierz pomyłek dla zbioru testowego i treningowego



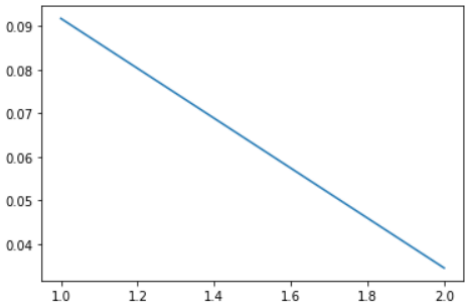
Przykłady cyfr, które zostały niepoprawnie sklasyfikowane.

Algorytm sprawdza się lepiej niż poprzednio omawiane rozwiązania jednak nie zawsze rozpoznaje cyfry poprawnie a najgorzej radzi sobie z rozpoznawaniem szóstek, tak jak pozostałe rozwiązania. Na macierzach pomyłek procent rozpoznanych cyfr sześć to 89% i 90% dla odpowiednio zbioru testowego i treningowego.

Transfer Sieci – uczenie całej sieci

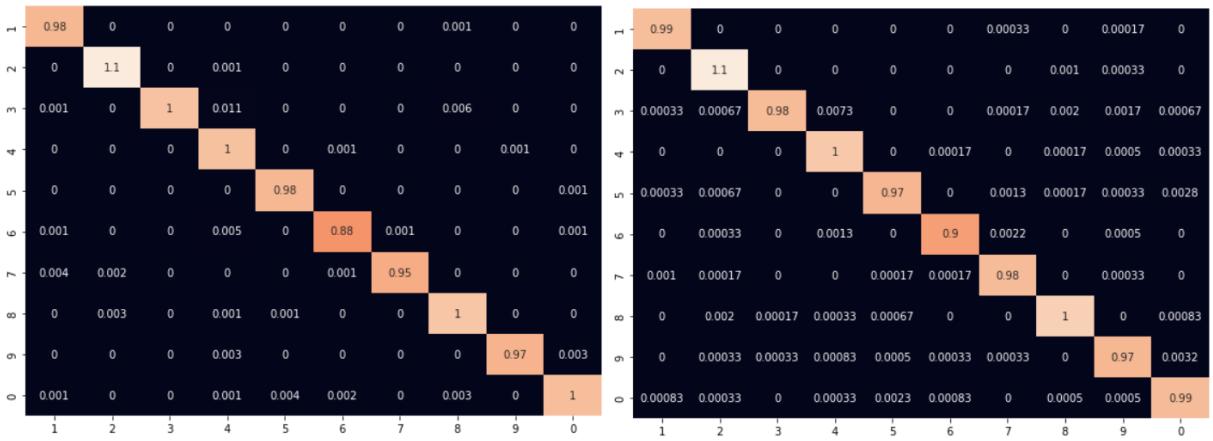
```
classifier = nn.Sequential(
    nn.Dropout(),
    nn.Linear(9216, 4096),
    nn.ReLU(),
    nn.Dropout(),
    nn.Linear(4096, 4096),
    nn.ReLU(),
    nn.Linear(4096, 10),
)
```

W tym wariancie została użyta sieć alexnet oraz własny klasyfikator jak dla poprzedniego eksperymentu. Składa się on z czterech warstw. Trzy z nich to warstwy liniowe po których wykonywana jest funkcja aktywacji ReLU. Pierwszą warstwą jest warstrwa dropout. Następnie wykonywane są wcześniej wspomniane warstwy liniowe.

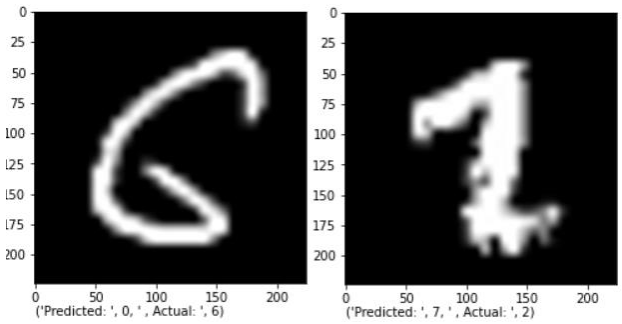


Użytą funkcją celu jest Cross Entropy Loss a optymalizatorem SDG. Nauka odbywała się dla dwóch epok.

Czas potrzebny do nauki wynosi ***307m 42,3s*** dla CPU: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz i pamięci RAM 16,00 GB. Skuteczność wyniosła 98,64%.



Macierz pomyłek dla zbioru testowego i treningowego



Przykłady cyfr, które zostały niepoprawnie sklasyfikowane.

Algorytm sprawdza się najlepiej z przedstawionych rozwiązań a najgorzej radzi sobie z rozpoznawaniem szóstek, tak jak pozostałe rozwiązania. Na macierzach pomyłek procent rozpoznanych cyfr sześć to 88% i 90% dla odpowiednio zbioru testowego i treningowego.