Dla liczby epok 2500 (najczęściej wykorzystywanej przeze mnie do testów w liście3) osiągnąłem następujące wyniki:

|  |  |  |  |
| --- | --- | --- | --- |
| Optimizer/batch size | 5 | 10 | 20 |
| SGD | Acc: 0.52  Prec: 0.0  F1: 0.0 | Acc: 0.573  Prec: 0.75  F1: 0.272 | Acc: 0.666  Prec: 0.703  F1: 0.603 |
| Adam | Acc: 0.84  Prec: 0.9  F1: 0.818 | Acc: 0.813  Prec: 0.866  F1: 0.787 | Acc: 0.786  Prec: 0.812  F1: 0.764 |
| RMSprop | Acc: 0.786  Prec: 0.8125  F1: 0.764 | Acc: 0.76  Prec: 0.781  F1: 0.735 | Acc: 0.746  Prec: 0.774  F1: 0.716 |

Powyższe wyniki zostały wyliczone dla wartości learning\_rate = 0,001. Dla większych wartości learning\_rate model nie był w stanie się uczyć.

Dla zadanych danych, najlepiej radzi sobie optimizer Adam, we wszystkich przypadkach(wartościach batch size) osiągał on najwyższe wyniki metryk. Najoptymalniejszą wartością batch\_size dla niego była wartość 5.

Przetestowałem również optimizery dla learning\_rate = 0,0001. Takie ustawienie odwraca nieco podium optimizerów:

|  |  |
| --- | --- |
| Adam Learning rate: 0.0001 Batch: 3  Acc: 0.84 Prec: 0.9 F1: 0.818  Adam Learning rate: 0.0001 Batch: 5  Acc: 0.826 Prec: 0.896 F1: 0.799  Adam Learning rate: 0.0001 Batch: 10  Acc: 0.826 Prec: 0.896 F1: 0.799  Adam Learning rate: 0.0001 Batch: 20  Acc: 0.826 Prec: 0.896 F1: 0.799 | RMSprop Learning rate: 0.0001 Batch: 3  Acc: 0.853 Prec: 0.903 F1: 0.835  RMSprop Learning rate: 0.0001 Batch: 5  Acc: 0.853 Prec: 0.903 F1: 0.835  RMSprop Learning rate: 0.0001 Batch: 10  Acc: 0.853 Prec: 0.903 F1: 0.835  RMSprop Learning rate: 0.0001 Batch: 20  Acc: 0.853 Prec: 0.903 F1: 0.835 |

W tym przypadku, najlepiej radzi sobie RMSprop(SGD nie uczy się przy tak niskim lr).