CNN d)

November 4, 2022

Para o item d) foi proposta uma camada convolucional adicional com uma redução na dimensão do Kernel, a fim de não reduzir tanto a dimensão dos feature maps resultantes da convolução. Além disso, o número de Kernels de cada camada foi aumentado para 15, para que mais feature maps fossem gerados. A camada de Pooling permaneceu sendo do tipo MaxPooling, contudo após a camada de Flatten foi feito um processo de Dropout. O Dropout consiste em atribuir uma probabilidade para que os neurônios sejam desativados a cada passo de treinamento. Isto contribui para que cada neurônio desempenhe um papel útil para a rede por conta própria, diminuindo a dependencia dos seus vizinhos. No caso desta atividade a probabilidade que resultou em um melhor desempenho foi de 0.01. Usando esta configuração a rede neural conseguiu atingir uma acurácia de 90% junto aos dados de teste em 20 épocas de treinamento, representando um ganho de 6% em relação aos itens c) e a).

[6]:
```
CNN = Sequential()

CNN.add(Conv2D(filters = 15,
               kernel_size = (2, 2),
               activation ='relu',
               input_shape = (28,28,3)))

CNN.add(Conv2D(filters = 15,
               kernel_size = (2, 2),
               activation ='relu',
               input_shape = (28,28,3)))


CNN.add(MaxPooling2D(pool_size=(5, 5)))

CNN.add(Flatten())

CNN.add(Dropout(0.01))

CNN.add(Dense(8, activation='softmax'))


CNN.compile(loss = 'categorical_crossentropy',
            optimizer = Adam(learning_rate = 1e-2),
            metrics = ['accuracy'])
```

```
CNN.fit(Train_Images, One_Hot_Train,
        batch_size = 100,
        epochs = 20,
        verbose = 0,
        validation_data=(Val_Images, One_Hot_Val))

Preds = []
Soft_Preds = CNN.predict(Test_Images, verbose = 0)
for Pred in Soft_Preds:
    Preds.append(np.argmax(Pred))

print("Acurácia:", accuracy_score(Preds, Test_Labels))

CM = confusion_matrix(Test_Labels, Preds, labels=[0, 1, 2, 3, 4, 5, 6, 7])
Formata_Matriz(CM)
```

```
Acurácia: 0.9017831043554516
```

[6]:

	Basófilos	Eosinófilos	Eritroblastos	\
Basófilos	215	1	1	
Eosinófilos	10	603	0	
Eritroblastos	4	1	288	
Granulócitos Imaturos	54	5	12	
Linfócitos	2	0	9	
Monócitos	11	0	2	
Neutrófilos	7	4	9	
Plaquetas	0	0	1	

	Granulócitos Imaturos	Linfócitos	Monócitos	\
Basófilos	18	2	7	
Eosinófilos	3	0	2	
Eritroblastos	10	2	1	
Granulócitos Imaturos	445	7	38	
Linfócitos	6	222	4	
Monócitos	44	2	224	
Neutrófilos	22	3	2	
Plaquetas	0	0	0	

	Neutrófilos	Plaquetas
Basófilos	0	0
Eosinófilos	6	0
Eritroblastos	3	2
Granulócitos Imaturos	18	0
Linfócitos	0	0
Monócitos	1	0
Neutrófilos	619	0

```
Plaquetas                       0         469
```

[5]:
```
i = 0
for Pred, Test_Label, Soft_Pred in zip(Preds, Test_Labels, Soft_Preds):
    if Pred != Test_Label and i <= 5:
        print("Classe Esperada:", Test_Label[0])
        print("Classe Predita:", Pred)
        print("Probabilidades", np.array(Soft_Pred))
        print("\n\n")
        i += 1
```

```
Classe Esperada: 1
Classe Predita: 3
Probabilidades [5.9318e-02 1.3811e-02 1.0516e-02 9.0688e-01 3.4214e-04
1.4075e-04
 8.9884e-03 4.2572e-15]



Classe Esperada: 1
Classe Predita: 0
Probabilidades [8.7337e-01 1.2010e-02 6.8291e-03 3.0490e-02 4.8740e-05
3.0909e-04
 7.6940e-02 6.0836e-11]



Classe Esperada: 2
Classe Predita: 3
Probabilidades [1.7396e-02 1.6811e-03 3.1182e-01 3.3009e-01 3.1264e-01
2.5037e-02
 1.3303e-03 1.4767e-09]



Classe Esperada: 3
Classe Predita: 0
Probabilidades [5.9438e-01 3.6710e-01 6.4155e-03 1.1037e-02 8.7633e-09
1.0978e-04
 2.0957e-02 4.0049e-16]



Classe Esperada: 3
Classe Predita: 0
Probabilidades [8.8257e-01 2.2950e-04 1.0444e-03 4.9080e-02 5.5609e-06
6.6325e-02
```

```
 7.4455e-04 1.6487e-15]



Classe Esperada: 5
Classe Predita: 0
Probabilidades [8.8638e-01 2.6673e-05 1.6109e-03 3.4855e-03 2.6557e-02
8.1800e-02
 1.3786e-04 5.2798e-11]
```