Monochrome Dreams

Sora Ioana-Georgiana, 342

1. Introduction

Given a number of monochromatic images, I had to use a multi-class classification algorithm to predict the class label of an image.

More about the project : https://www.kaggle.com/c/ai-unibuc-24-22-2021/overview

2. Data exploring & preprocessing

We are given 3 sets of data, one for training, one for validation and the last one for testing, having the class labels for each set, except the testing one (where we have to predict its class label).

I chose to read the files using Pandas library(where I ignored the header for each file and used the comma separator to separate the name of the image of its label), obtaining the following table:

	Data	Label
0	000000.png	6
1	000001.png	0
2	000002.png	0
3	000003.png	5
4	000004.png	8
...	...	...
29996	029996.png	0
29997	029997.png	3
29998	029998.png	0
29999	029999.png	5
30000	030000.png	0

30001 rows × 2 columns

As it's shown, the images aren't given directly, so we have to read them from the folder, using keras.preprocessing.image. Here we have some steps to follow for each image:

1. We load the image using load_img with the parameters : directory + name_of_file and the desired size, in my case: 28x28x3
2. We transform it to an array using img_to_array

3. Then we make sure that every pixel has the type of float, so we won't lose data while doing the next step.
4. Knowing that the maximum value of a pixel can be 255, we divide each pixel by 255, so the ML algorithm will classify the images with more accuracy
5. We add every image to a numpy array.

Then, we use the same algorithm for the validation and testing sets.

Also, for the class labels, I used Pandas library' method get_dummies, to transform the value of each class to an array.

E.g.: For 9 classes and a label of 3, the array would look like : [0, 0, 0, 1, 0, 0, 0, 0, 0]

3. Training

I chose to use a Convolutional Neural Network model, because it represents the data as grid structures, being the reason why it works well for image classification problems.

I created the model, using the following steps:

1. Insert an input 2D convolution layer with the parameters :

- kernel_size = (3,3), the height and width of the 2D convolution window

-padding = same, padding evenly up/down or left/right, so the output has the same dimensions as the input

- filters = 32, meaning the number of output filters

- activation = 'tanh', a hyperbolic tangent activation function that ranges between (-1 to 1)

- input_shape=(28, 28, 3), meaning the size of each image

2. Insert another 2D convolution layer, using almost the same parameters, excluding the input_shape
3. Insert a pooling layer, using MaxPool2D of size (2,2)
4. Repeat steps 2&3. And again step 2.
5. Add a Flatten layer, that flattens the input.
6. Add 3 Dense layers, the first ones are using a Rectified linear unit activation(usually used for computer vision applications, as in our case for an image classification). And the third layer is used as an output layer, of parameters number of classes(9) and an activation of type Softmax.

After adding all the layers, we're compiling the model using the parameters:

-loss = Categorical Crossentropy, which computes the loss between the labels and the predictions

-metrics = Accuracy, because as the model is training, we want to see the accuracy of the model.

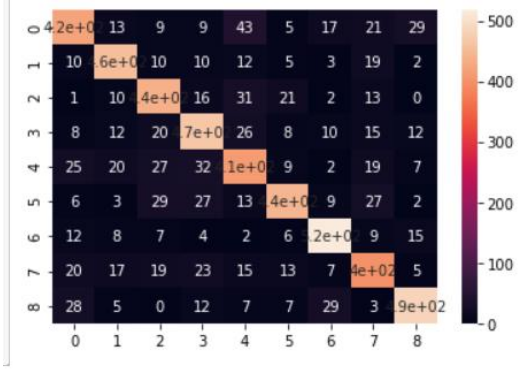
-optimizer = Adam. Most of the optimizers are based on the gradient descent algorithm, I chose this one because it works well on large datasets.

Finally, to train the model I chose to do it on 20-25 epochs, so it won't over-fit the CNN model.

4. Evaluation

In order to evaluate the model I am considering the following measures : accuracy score, f1 score, confusion matrix and classification report.

Confusion matrix :



Classification report:

	precision	recall	f1-score	support
0	0.79	0.74	0.77	570
1	0.84	0.87	0.85	527
2	0.78	0.82	0.80	533
3	0.78	0.81	0.79	578
4	0.73	0.75	0.74	554
5	0.86	0.79	0.82	561
6	0.87	0.89	0.88	580
7	0.76	0.77	0.77	520
8	0.87	0.84	0.86	577
accuracy			0.81	5000
macro avg	0.81	0.81	0.81	5000
weighted avg	0.81	0.81	0.81	5000

5. Comparison with other models

I have also tried to run other models, to check if the accuracy improves, but the answer was negative.

First I reshaped the input size of the images, given the fact that most of the classifiers don't accept 4 dimensional arrays and then I used the classifiers with basic parameters.

1. Decision Tree

Accuracy: 0.36

Classification report:

Classification report :

	precision	recall	f1-score	support
0	0.27	0.25	0.26	570
1	0.38	0.39	0.38	527
2	0.32	0.33	0.32	533
3	0.37	0.35	0.36	578
4	0.30	0.30	0.30	554
5	0.45	0.45	0.45	561
6	0.44	0.43	0.43	580
7	0.31	0.33	0.32	520
8	0.45	0.46	0.45	577
accuracy			0.37	5000
macro avg	0.36	0.36	0.36	5000
weighted avg	0.37	0.37	0.37	5000

2. Random Forest

Accuracy: 0.62

Classification report:

Classification report :

	precision	recall	f1-score	support
0	0.54	0.49	0.51	570
1	0.63	0.69	0.66	527
2	0.60	0.57	0.58	533
3	0.65	0.61	0.63	578
4	0.56	0.57	0.56	554
5	0.70	0.67	0.68	561
6	0.66	0.72	0.69	580
7	0.55	0.54	0.55	520
8	0.68	0.73	0.70	577
accuracy			0.62	5000
macro avg	0.62	0.62	0.62	5000
weighted avg	0.62	0.62	0.62	5000

3. SGD

Accuracy: 0.55

Classification report:

```
Classification report :
              precision    recall  f1-score   support

           0       0.40      0.55      0.46       570
           1       0.50      0.66      0.57       527
           2       0.73      0.44      0.55       533
           3       0.53      0.70      0.60       578
           4       0.52      0.42      0.47       554
           5       0.80      0.45      0.58       561
           6       0.60      0.59      0.59       580
           7       0.65      0.44      0.52       520
           8       0.53      0.68      0.60       577

    accuracy                           0.55      5000
   macro avg       0.58      0.55      0.55      5000
weighted avg       0.58      0.55      0.55      5000
```

Final results :

	Precision	Recall	f1	Accuracy
CNN	0.81	0.81	0.81	0.8096
---	---	---	---	---
Decision Tree	0.36	0.36	0.36	0.36
---	---	---	---	---
Random Forest	0.62	0.62	0.62	0.6208
---	---	---	---	---
SGD	0.58	0.55	0.55	0.55

Which proves that CNN is the best model to use in this project.