Image Cleansing using Morphology and Deep Learning

Munkhsukh Munkhtsetseg

Eötvös Loránd University, H-1053 Budapest, Egyetem tér 1-3
https://www.elte.hu/

Abstract. The goal of this paper is to contribute to image cleansing for the pre-processing. To achieve, we use a several techniques of Morphological filters and sharpening. However, choosing the best technique amongst them is always challenging. Hence we will use Deep Learning, specially Convolutional Neural Network, to find best a filter specific input image.

Keywords: Image cleansing · Morphology filters · Convolutional Neural Network.

1 Introduction

In image processing projects, the image cleansing can improve significantly our accuracy and results. Hence, it consumes much time and efforts to determine appropriate image filters. Our work aim to help this step. Here we provide short introduction of Morphology Filters and sharpening.

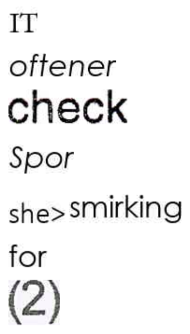


Fig. 1: Preview. Sample images from the data set

1.1 Data set

The dataset from kaggle.com has 10k images from the main dataset. This dataset is a mix bag of images gathered from multiple sources. Manually cropped and labelled images from natural scanned documents. Synthetically generated images which look very similar to natural images to boost infrequent characters. Data labelled using tesseract OCR software and manually checked for OCR errors.

1.2 Exploratory of the Dataset

Images are in raw format and do not have a specific size (Fig 1). Firstly, let us glance at the most frequent words. As we see (Table 1), stopwords, short function words, are in common.
In the data set, the words containing special characters: 1440 (14%) and 4251 unique words (Fig!2).

the	504
of	297
and	204
to	170
in	128
is	89
a	73
are	62
THE	59
What	58
for	58
Explain	55

Table 1: The most frequent words

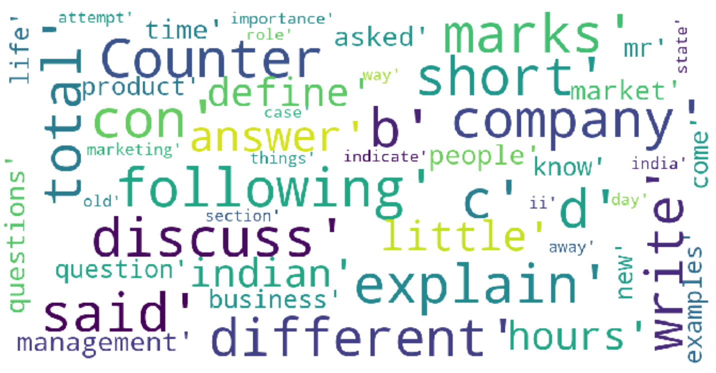


Fig. 2: Wordcloud.

2 Morphology filters

Morphological transformations are some simple operations based on the image shape. It needs two inputs, one is our original image, second one is called structuring element or kernel which decides the nature of operation.Two basic morphological operators are Erosion and Dilation. Then its variant forms like Opening, Closing, Gradient etc also comes into play. [1]

Dilate and Erode The basic idea of erosion is just like soil erosion only, it erodes away the boundaries of foreground object. So what does it do? The kernel slides through the image. A pixel in the original image will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded.
It is just opposite of erosion. Here, a pixel element is '1' if at least one pixel under the kernel is '1'. So it increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object.

Opening and Closing Opening is just another name of erosion followed by dilation. It is useful in removing noise, as we explained above.
Closing is reverse of Opening, Dilation followed by Erosion. It is useful in closing small holes inside the foreground objects, or small black points on the object.

Structuring Element / Kernel We manually created a structuring elements in the previous examples with help of Numpy. It is rectangular shape. But in some cases, you may need elliptical/circular shaped kernels. In below image, we can see simple 5 by 5 kernel shapes (Fig 2).

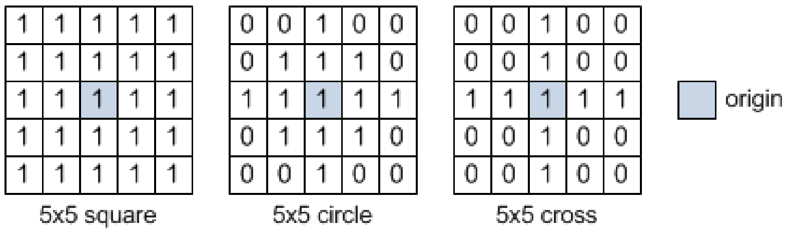


Fig. 3: Simple structuring elements.

2.1 Challenge

In our practice, finding proper kernel size and shape is challenging. Hence we did some experiment to find it. We modified the following parameters of the morphology operations and examined how it influenced the OCR prediction using tesseract OCR.

- Kernel size: 2 to 6
- Kernel type: rectangle, cross and ellipse
- Number iteration: 1 to 4

Moreover, we combined some basic operation over and over such as opening after closing and indicates OC, O means opening and C means closing. Here all operation we used in our experiment.

- Erosion
- Dilation
- Opening
- Closing
- OC
- CO
- OCO
- COC

Each operations with each their parameters filtered over first fifty images. Then, they were labeled by OCR tesseract and calculated percentages of correct labels.

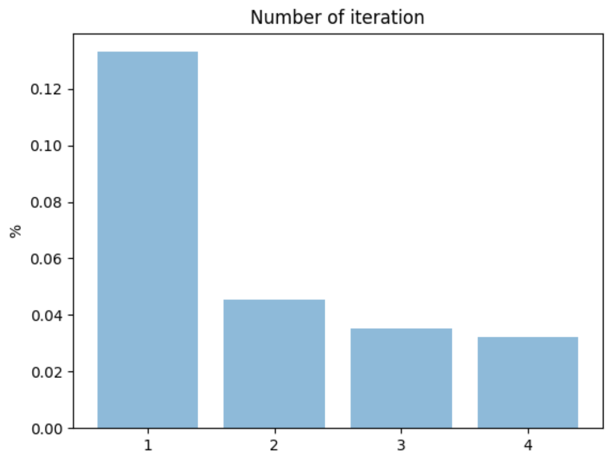


Fig. 4: The correct labelled images percent of the first 50 images grouped by iteration number

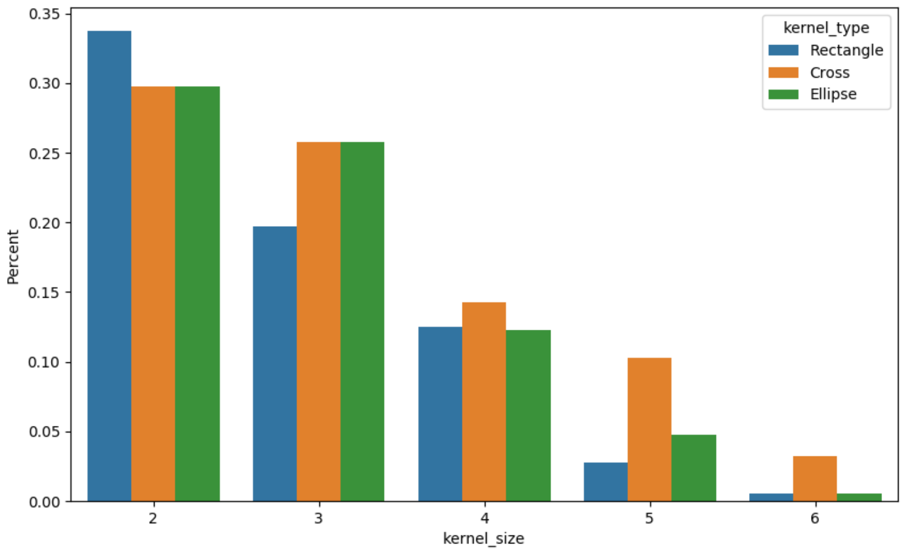
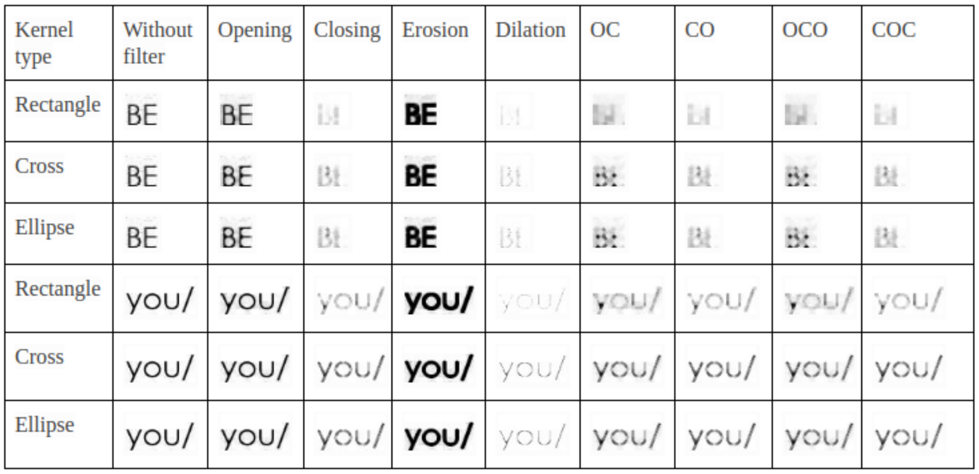


Fig. 5: The correct labelled images percent of the first 50 images grouped by kernel size



Kernel type	Without filter	Opening	Closing	Erosion	Dilation	OC	CO	OCO	COC
Rectangle	BE	BE	[illegible]	BE	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Cross	BE	BE	[illegible]	BE	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Ellipse	BE	BE	[illegible]	BE	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Rectangle	you/	you/	you/	you/	you/	you/	you/	you/	you/
Cross	you/	you/	you/	you/	you/	you/	you/	you/	you/
Ellipse	you/	you/	you/	you/	you/	you/	you/	you/	you/

Fig. 6: An example of comparing of operations

Here we found out too much iteration and kernel size can not improve OCR performance (Fig 4, 5). Hence we changed the ranges of kernel size and number iterations to 2 to 3 and 1 to 2 respectively.

Then, We did our experiment over all ten thousands with the mentioned parameters.

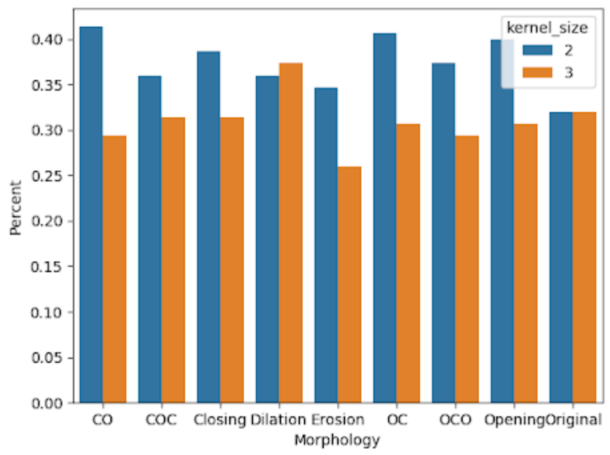


Fig. 7: The correct labelled images percent of the whole dataset grouped by kernel size and operations

From the graph (Fig 7), we can see kernel size 2 is significantly good results with all operations except the dilation operation. Also, each kernel type has almost the same results.

Sharpening after blurring We also some tried some other filtering techniques such as otsu, histogram equalizer and the their combinations. Sharpening after blurring gave the better result.

Image blurring is achieved by convolving the image with a low-pass filter kernel. It is useful for removing noise. It actually removes high frequency content (eg: noise, edges) from the image[2].

We used averaging filter with the kernel 3 by 3 filled by ones.

After that, Image sharpen operation is used to increase the edges distance (Fig 9).

The proper kernel size and type Now, we can choose the proper one with from the results. As we know, more iteration number made it worse image. Hence we chose it would be 1. From Table 2, kernel type is Cross and kernel size is 2 are the most result.

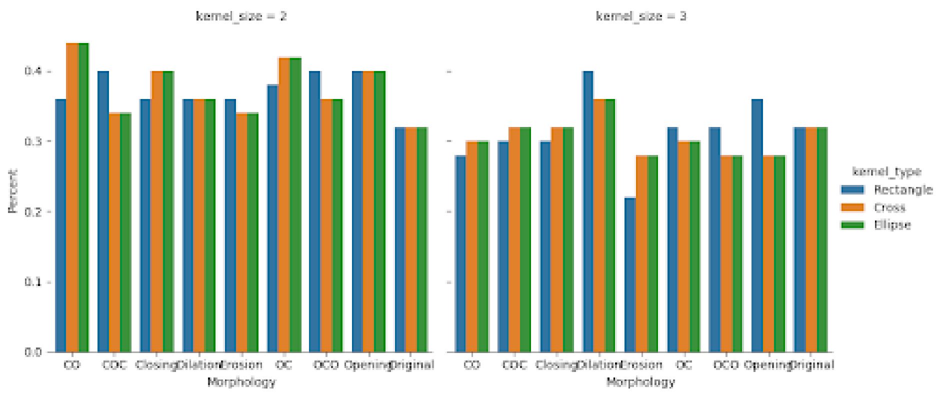
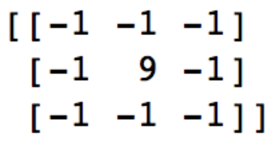


Fig. 8: The correct labelled images percent of the whole dataset grouped by kernel size, kernel type and operations



```
[[-1 -1 -1]
 [-1  9 -1]
 [-1 -1 -1]]
```

Fig. 9: Sharpening kernel

kernel_type	kernel_size	Count
Rectangle	**2**	4387.000000
	3	4018.444444
Cross	**2**	4511.666667
	3	4072.333333

Table 2: The aggregated results by kernel type and kernel size

3 CNN implementation

Our task is supervised machine learning task. We will use Deep learning with Convolutional Neural Network architecture.

3.1 Labeling

Simple approach Firstly, we need to define output labels which is our operations to determine each images assigned a proper operation to remove noise and to prepare OCR operations. From Table 3, There can be more than one operations can be our output label. Hence, first, we did a simple and naive approach, which is to combine all operations if predicted correctly. For instance, if only opening and closing has predict correct our output label will be an array of opening and closing. After finding all possible 176 combinations from the result table, we encoded it to numbers to 0 to 175.

Choose one label by random choice Apparently, We have too many labels, making it complex. To reduce them, as we know we had already the array of operations, however it is enough only one operation.Hence we can choose only one operation by randomly. Now, we have only ten output labels as the same number of our operations.

Image	1044.jpeg	1045.jpeg	1046.png	1047.jpeg	1048.jpeg	1049.jpeg	105.jpeg	1050.jpeg	1051.png
Actual word	Lion	the	apple	better	without	it	they	em	Western
Kernel size	2	2	2	2	2	2	2	2	2
Kernel type	0	0	0	0	0	0	0	0	0
Iterations	1	1	1	1	1	1	1	1	1
Without Filter	Lion			better	without	it	they	em	
Opening	Lion	the		better	without	it	they	em	Western
Closing	Lion	the		better	without	it	they	em	Western
Erosion	Lion			better	without	it	they		
Dilation	Lion	the		better	without	it	they	erm	Western
OC	Lion	the	apple	better	without		they	erm	Weastern
CO	Lion	the	apple	better	without		they	erm	Weastern
OCO	Lion	the	apple	better	without		they	err	Weeastern
COC	Lion	the	apple	better	without		they	err	Weastern
Sharpened	Lion	the		better	without	it	they	em	

Table 3: Samples from the result table

In the next steps, we will examine both output label sets.

3.2 CNN models

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernels), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values [3]. Fist of all, CNN models performs the same dimensional image and therefore we resized all images to the mean dimension: (33, 92)
Then, We split the data set to train(80%) and test set(20%).

Now it is time to define our model. For simplicity, We defined the following a CNN model.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param 
    #
=================================================================
conv2d_1 (Conv2D)            (None, 31, 90, 8)         224
_________________________________________________________________
average_pooling2d_1 (Average (None, 15, 45, 8)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 13, 43, 16)        1168
_________________________________________________________________
average_pooling2d_2 (Average (None, 6, 21, 16)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2016)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               201700
_________________________________________________________________
dense_2 (Dense)              (None, 80)                8080
_________________________________________________________________
dense_3 (Dense)              (None, 176)               14256
=================================================================
Total params: 225,428
Trainable params: 225,428
```

```
Non-trainable params: 0
```

Listing 1.1: Simple CNN model

The optimization method is 'Adam' and Loss function is Sparse categorical crossentropy which can be used if the targets are numbers [4].

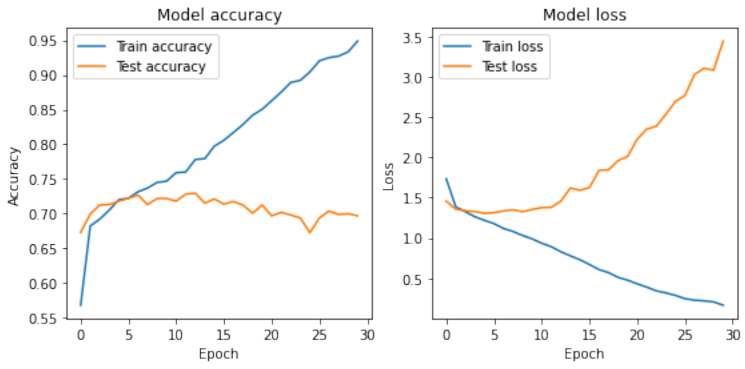


Fig. 10: Plot training validation accuracy values

Then, we fit our model with the label set which randomly chose (Fig 13)

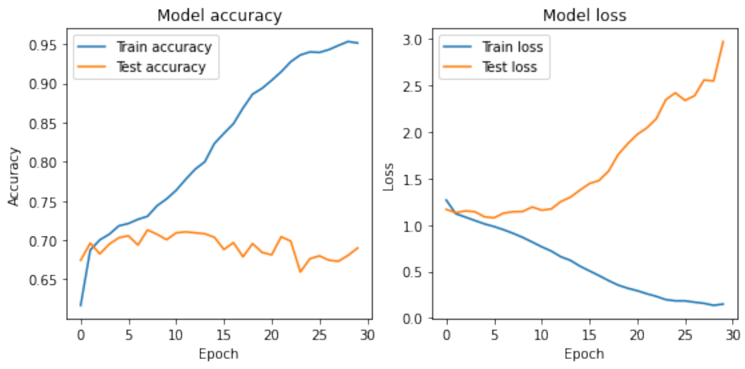


Fig. 11: Plot training validation accuracy values with the label set randomly chose

Here, we can see a overfitting in our model because train loss is high but test loss is low.

Evaluation of the model Now, we will evaluate our model with our test data set: Test loss: 3.49, Test accuracy: 70.70%. And, evaluation of dataset with the labels with randomly chose: Test loss: 2.84, Test accuracy: 67.70%.

Our next step is improve the model and avoid the overfitting.

3.3 Improvement

Firstly, In order to avoid the overfitting we can put Dropout in the CNN model. During training, some number of layer outputs are randomly ignored or “dropped out.” This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different “view” of the configured layer.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param
    #
=================================================================
conv2d_5 (Conv2D)            (None, 31, 90, 8)         224
_________________________________________________________________
average_pooling2d_5 (Average (None, 15, 45, 8)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 13, 43, 16)        1168
_________________________________________________________________
average_pooling2d_6 (Average (None, 6, 21, 16)         0
_________________________________________________________________
flatten_3 (Flatten)          (None, 2016)              0
_________________________________________________________________
dense_7 (Dense)              (None, 100)               201700
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_8 (Dense)              (None, 80)                8080
_________________________________________________________________
dropout_2 (Dropout)          (None, 80)                0
_________________________________________________________________
dense_9 (Dense)              (None, 176)               14256
```

```
=================================================================
Total params: 225,428
Trainable params: 225,428
Non-trainable params: 0
```

Listing 1.2: Simple CNN model with Dropout

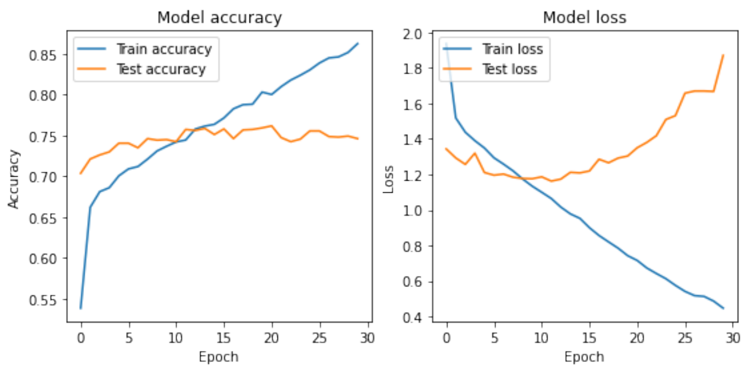


Fig. 12: With dropout. Plot training validation accuracy values

Then, we fit our model with the label set which randomly chose (Fig **??**)

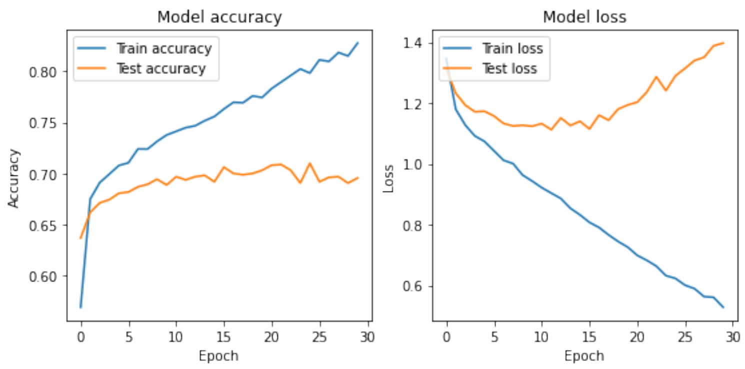


Fig. 13: With dropout. Plot training validation accuracy values with the label set randomly chose

Evaluation of the model Now, we will evaluate our model with our test data set: Test loss: 2.33, Test accuracy: 74.55%. And, evaluation of dataset with the labels with randomly chose: Test loss: 1.34, Test accuracy: 70.45%.

To compare last model our loss is decreased and test accuracy increased.

We used before 2 layers CNN. Now we go deeper, 3 layers architecture.

```
Layer (type)                 Output Shape              Param
    #
=================================================================
conv2d_25 (Conv2D)           (None, 31, 90, 8)         224
_________________________________________________________________
average_pooling2d_25 (Averag (None, 15, 45, 8)         0
_________________________________________________________________
dropout_32 (Dropout)         (None, 15, 45, 8)         0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 13, 43, 16)        1168
_________________________________________________________________
average_pooling2d_26 (Averag (None, 6, 21, 16)         0
_________________________________________________________________
dropout_33 (Dropout)         (None, 6, 21, 16)         0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 4, 19, 32)         4640
_________________________________________________________________
average_pooling2d_27 (Averag (None, 2, 9, 32)          0
_________________________________________________________________
dropout_34 (Dropout)         (None, 2, 9, 32)          0
_________________________________________________________________
flatten_12 (Flatten)         (None, 576)               0
_________________________________________________________________
dense_34 (Dense)             (None, 100)               57700
_________________________________________________________________
dropout_35 (Dropout)         (None, 100)               0
_________________________________________________________________
dense_35 (Dense)             (None, 80)                8080
_________________________________________________________________
dropout_36 (Dropout)         (None, 80)                0
```

```
_________________________________________________________________
dense_36 (Dense)             (None, 11)                891
=================================================================
Total params: 72,703
Trainable params: 72,703
Non-trainable params: 0
_________________________________________________________________
```

Listing 1.3: 3 layers CNN model

Test loss: 1.04 and Test accuracy: 72.00%.

4 Conclusion

In this paper, I proposed to image cleansing using Morphology filters and find proper operation for a specific image using CNN model. We reached our goal and our model has 72 percent accuracy.

References

1. Morphological Transformations Last accessed 25 May 2020 https://opencv-python-tutroals.readthedocs.io/en/latest/py$_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html$
2. Image filtering
Last accessed 25 May 2020 https://docs.opencv.org/master/d4/d13/tutorial$_{py}filtering.html$
3. Understanding of Convolutional Neural Network (CNN) — Deep Learning
Last accessed 25 May 2020 https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148
4. Diederik P. Kingma, Jimmy Ba, Method for Stochastic Optimization