

Capstone Project Report

Denoising Images using Autoencoders

Machine Learning Engineer Nanodegree

Pushkar Kurhekar
2-1-2021

I. Definition

Project Overview

For Optical Character Recognition (OCR) tasks, quality of the input document is directly proportional to the output accuracy of the text. Quality problems in the input document images such as skew, perspective transformation, watermarks and noise can severely reduce the accuracy of these OCR models. In terms of noise, random dust particles can be seen in old flatbed scanners which are not maintained correctly, or if old books are being scanned, we may see folded or yellowed pages, stained pages, etc. All these problems lead to worse output when used for automating of information retrieval / extraction from a pipeline of scanned document images.

Problem Statement

The problem that is being tackled here is the task of noise removal (denoising) from scanned document images. An autoencoder neural network is used for this task. It is trained on the noisy and cleaned versions of the documents and is tested on previously unseen noisy documents. With additional data and variety in the noise patterns and shapes, this model can be extended to a larger variety of noise in documents as required.

Metrics

Mean Square Error is used for evaluation as a loss function for the Autoencoder model. For the generated output files, Root Mean Square Error (RMSE) is used, as this is being used by the Kaggle competition on submission of results. Each pixel of the output image is compared to the testing set which is used by the competition.

RMSE is defined as follows:

$$RMSE = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}}$$

$\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$ are predicted values
 y_1, y_2, \dots, y_n are observed values
 n is the number of observations

Image credits: <https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e>

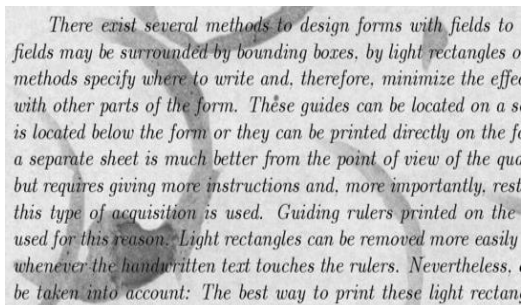
Since the errors are squared before they are averaged, the RMSE gives high weightage to larger errors.

II. Analysis

Data Exploration & Visualization

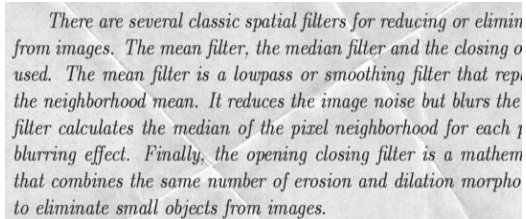
The dataset contains 144 noisy training samples, 144 clean training samples and 72 test samples. It is available at the Kaggle Competition link [here](#), and also available at the UCI Machine Learning Repository [here](#).

A pattern in the training samples that was noticed is that the noise belongs to one of 6 categories. They are shown below along with a description.



There exist several methods to design forms with fields to fields may be surrounded by bounding boxes, by light rectangles o methods specify where to write and, therefore, minimize the effect with other parts of the form. These guides can be located on a sheet of paper that is located below the form or they can be printed directly on the form. The use of guides on a separate sheet is much better from the point of view of the quality of the scanned image, but requires giving more instructions and, more importantly, restricts its use to tasks where this type of acquisition is used. Guiding rulers printed on the form are more commonly used for this reason. Light rectangles can be removed more easily with filters than dark lines whenever the handwritten text touches the rulers. Nevertheless, other practical issues must be taken into account: The best way to print these light rectangles

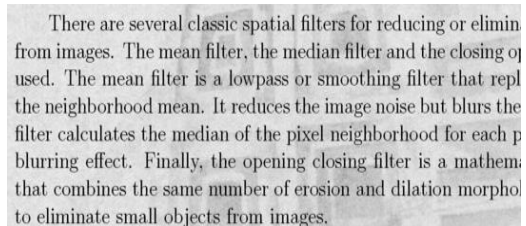
1. Stain



There are several classic spatial filters for reducing or eliminating noise from images. The mean filter, the median filter and the closing operation are commonly used. The mean filter is a lowpass or smoothing filter that replaces each pixel with the neighborhood mean. It reduces the image noise but blurs the image. The median filter calculates the median of the pixel neighborhood for each pixel, which reduces the blurring effect. Finally, the opening closing filter is a mathematical operation that combines the same number of erosion and dilation morphological operations to eliminate small objects from images.

The main goal was to train a neural network in a supervised learning task to clean a noisy image from a noisy one. In this particular case, it was much easier to clean a noisy image from a clean one than to clean a subset of noisy

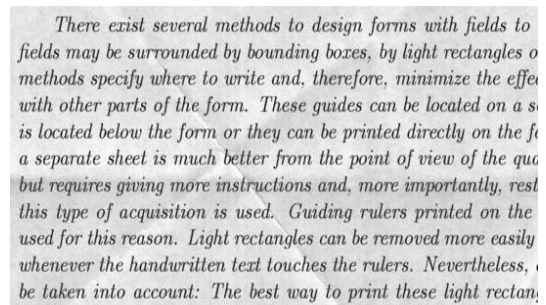
3. Double Folded Paper



There are several classic spatial filters for reducing or eliminating noise from images. The mean filter, the median filter and the closing operation are commonly used. The mean filter is a lowpass or smoothing filter that replaces each pixel with the neighborhood mean. It reduces the image noise but blurs the image. The median filter calculates the median of the pixel neighborhood for each pixel, which reduces the blurring effect. Finally, the opening closing filter is a mathematical operation that combines the same number of erosion and dilation morphological operations to eliminate small objects from images.

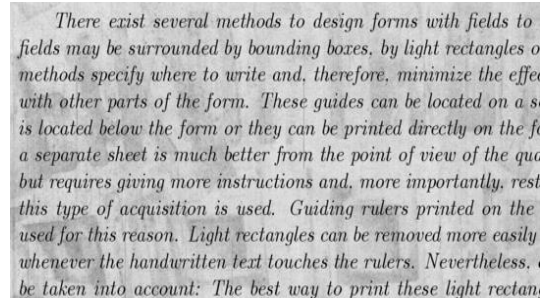
The main goal was to train a neural network in a supervised learning task to clean a noisy image from a noisy one. In this particular case, it was much easier to clean a noisy image from a clean one than to clean a subset of noisy

5. Background content variation



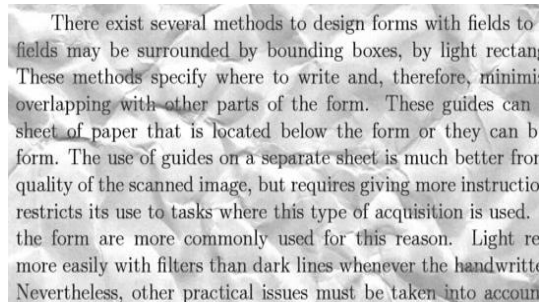
There exist several methods to design forms with fields to fields may be surrounded by bounding boxes, by light rectangles o methods specify where to write and, therefore, minimize the effect with other parts of the form. These guides can be located on a sheet of paper that is located below the form or they can be printed directly on the form. The use of guides on a separate sheet is much better from the point of view of the quality of the scanned image, but requires giving more instructions and, more importantly, restricts its use to tasks where this type of acquisition is used. Guiding rulers printed on the form are more commonly used for this reason. Light rectangles can be removed more easily with filters than dark lines whenever the handwritten text touches the rulers. Nevertheless, other practical issues must be taken into account: The best way to print these light rectangles

2. Folded Paper



There exist several methods to design forms with fields to fields may be surrounded by bounding boxes, by light rectangles o methods specify where to write and, therefore, minimize the effect with other parts of the form. These guides can be located on a sheet of paper that is located below the form or they can be printed directly on the form. The use of guides on a separate sheet is much better from the point of view of the quality of the scanned image, but requires giving more instructions and, more importantly, restricts its use to tasks where this type of acquisition is used. Guiding rulers printed on the form are more commonly used for this reason. Light rectangles can be removed more easily with filters than dark lines whenever the handwritten text touches the rulers. Nevertheless, other practical issues must be taken into account: The best way to print these light rectangles

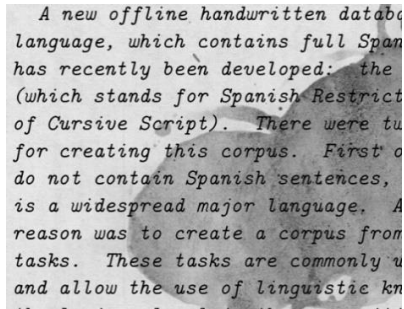
4. Background content



There exist several methods to design forms with fields to fields may be surrounded by bounding boxes, by light rectangles o methods specify where to write and, therefore, minimize the effect with other parts of the form. These guides can be located on a sheet of paper that is located below the form or they can be printed directly on the form. The use of guides on a separate sheet is much better from the point of view of the quality of the scanned image, but requires giving more instructions and, more importantly, restricts its use to tasks where this type of acquisition is used. Guiding rulers printed on the form are more commonly used for this reason. Light rectangles can be removed more easily with filters than dark lines whenever the handwritten text touches the rulers. Nevertheless, other practical issues must be taken into account: The best way to print these light rectangles

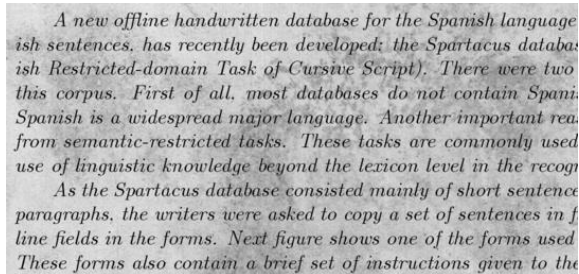
6. Crumpled Paper

The test set on the other hand contains 4 categories. The ‘Double Folded Paper’ and ‘Crumpled Paper’ categories seem to be identical, whereas the ‘Stain’ and ‘Background content’ have a different shape and pattern in the test data. In this case, the ‘Background content’ variation can be considered more like the ‘Stain’ category. They are shown in the table below:



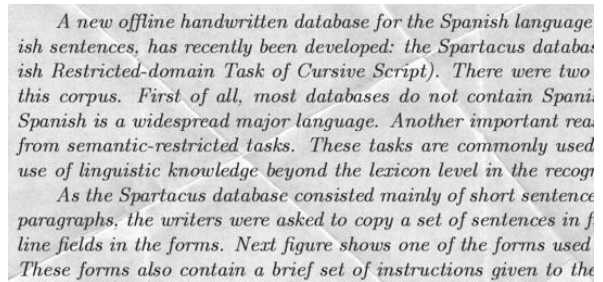
A new offline handwritten database for the Spanish language, which contains full Spanish sentences, has recently been developed: the Spartacus database (which stands for Spanish Restricted-domain Task of Cursive Script). There were two reasons for creating this corpus. First of all, most databases do not contain Spanish sentences, and Spanish is a widespread major language. Another important reason was to create a corpus from semantic-restricted tasks. These tasks are commonly used to allow the use of linguistic knowledge beyond the lexicon level in the recognition of handwritten text.

1. Stain Variation



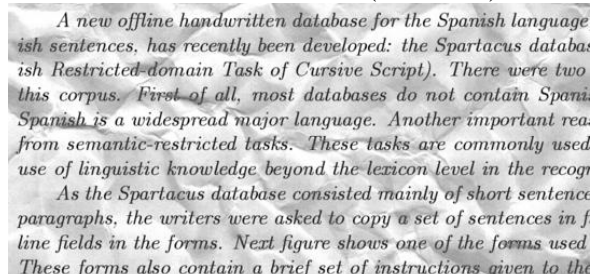
A new offline handwritten database for the Spanish language, which contains full Spanish sentences, has recently been developed: the Spartacus database (which stands for Spanish Restricted-domain Task of Cursive Script). There were two reasons for creating this corpus. First of all, most databases do not contain Spanish sentences, and Spanish is a widespread major language. Another important reason was to create a corpus from semantic-restricted tasks. These tasks are commonly used to allow the use of linguistic knowledge beyond the lexicon level in the recognition of handwritten text.

3. Background content variation



A new offline handwritten database for the Spanish language, which contains full Spanish sentences, has recently been developed: the Spartacus database (which stands for Spanish Restricted-domain Task of Cursive Script). There were two reasons for creating this corpus. First of all, most databases do not contain Spanish sentences, and Spanish is a widespread major language. Another important reason was to create a corpus from semantic-restricted tasks. These tasks are commonly used to allow the use of linguistic knowledge beyond the lexicon level in the recognition of handwritten text.

2. Double Fold (Identical)



A new offline handwritten database for the Spanish language, which contains full Spanish sentences, has recently been developed: the Spartacus database (which stands for Spanish Restricted-domain Task of Cursive Script). There were two reasons for creating this corpus. First of all, most databases do not contain Spanish sentences, and Spanish is a widespread major language. Another important reason was to create a corpus from semantic-restricted tasks. These tasks are commonly used to allow the use of linguistic knowledge beyond the lexicon level in the recognition of handwritten text.

4. Crumpled Paper (Identical)

Algorithms and Techniques

There are two main approaches for noise removal in document images.

i. Traditional approach













The traditional approach involves noise removal with the help of various morphological operations along with kernel-based methods such as Gaussian blur. This approach works well when the noise in the image is along the lines of salt-and-pepper noise [6]. The type of noise being dealt with here is more realistic in terms of the shape, size and intensity of the noisy pixels in the image. In the dataset as shown in the previous sections, the various noise types are not compatible with these approaches. In many cases, there is a good chance that the text in the image would get distorted or negatively affected, which would negate the entire purpose of the project, which is to maintain OCR accuracy.

ii. Neural Networks

Autoencoders, due to the compression of image data and its reconstruction, are especially good at detecting anomalies in images. Noise in the image is also a type of anomaly – typically noise is random pixels which contain no useful information for the image content. With this logic, autoencoders can be used to remove noise from the image if the neural network learns what the relevant content in the image is, which it would then reconstruct.

Benchmark

The Kaggle competition Leaderboard contains the various RMSE scores for submissions when the competition was active. A screenshot of the Leaderboard is as follows:

#	Δ pub	Team Name	Notebook	Team Members	Score 	Entries	Last
1	—				0.00416	5	5y
2	—	Colin			0.00512	29	5y
3	—	ironbar			0.01122	30	5y
4	—	Altexsoft Team			0.01318	4	5y
5	—	ShadowNet			0.01347	30	5y
6	—	toshi_k			0.01521	12	5y
7	—	Cameron McKenzie			0.01624	18	6y
8	—	nagadomi			0.01685	6	6y
9	—	Richard Weiss			0.01738	12	5y
10	—	zxytim			0.01794	27	5y

Kaggle Competition Leaderboard

We can see here that the top 10 results are all under 0.017 RMSE score. Due to this being a competition, the exact algorithms and / or models the participants used are unknown, so the only comparison to be made is based on the RMSE score itself. The 10th place score of 0.01794 is taken as a benchmark.

III. Methodology

Data Preprocessing

All the training images are read into memory, then converted into NumPy arrays, then normalized by dividing all pixel values by 255. This is done as the model works best with values between 0 and 1, and the pixel values in the images range from 0 to 255, 0 being black and 255 being white.

Image augmentation using the `imgaug` package was attempted, however the results were significantly worse. This might possibly be due to the model being confused on the noisy and cleaned augmented images, and therefore could not reconstruct the image correctly.

Implementation & Refinement

An Autoencoder Neural Network implemented in Tensorflow 2.x using Keras was used as the model of choice for denoising. The entire code was executed on Google Colab with a GPU runtime. During model development the following points were taken into consideration for increasing the accuracy:

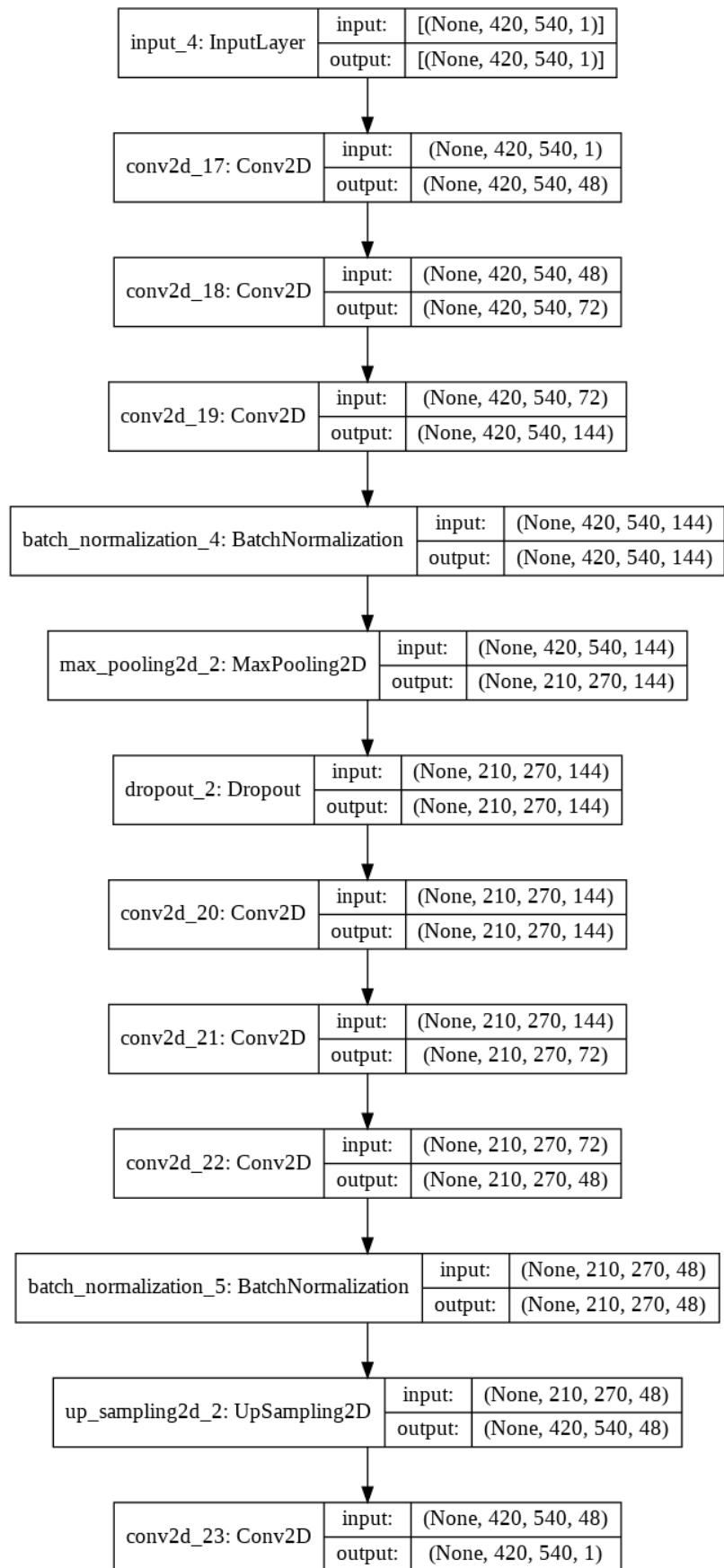
- Model architecture – This was decided after some experimentations and after taking a look at a few online references and tutorials [1-5]. Since the approach in all of these were for fairly small images, like the ones in MNIST or Fashion MNIST, I tried applying more convolution layers so that the model could extract more information and hopefully learn a bit better. This seems to have worked well.
- Batch normalisation – This was added for both the encoder and decoder parts of the model after the set of convolutional layers – this helped to improve the model accuracy and robustness.
- Dropout – A dropout of 50% was used to avoid overfitting on the input dataset. However, since the input data is very small, the model will still need to be retrained for noise removal on other types of noise in document images.
- Loss function choice – Mean Square Error is used as the loss function, since the Kaggle competition makes use of Root Mean Square Error (RMSE) for testing and placing the submission on the leaderboards.
- Optimization algorithm choice – The Adam optimizer was the most effective at minimizing the loss as compared to other optimizers. The default hyperparameters are being used since they work well with this dataset.

The model was trained for 200 epochs with a batch size of 16. The model summary and architecture are as follows:

Model: "model"

Layer (type)	Output Shape	Param #
=====		
input_3 (InputLayer)	[(None, 420, 540, 1)]	0
conv2d_10 (Conv2D)	(None, 420, 540, 48)	480
conv2d_11 (Conv2D)	(None, 420, 540, 72)	31176
conv2d_12 (Conv2D)	(None, 420, 540, 144)	93456
batch_normalization_2 (Batch Normalization)	(None, 420, 540, 144)	576
max_pooling2d_1 (MaxPooling2D)	(None, 210, 270, 144)	0
dropout_1 (Dropout)	(None, 210, 270, 144)	0
conv2d_13 (Conv2D)	(None, 210, 270, 144)	186768
conv2d_14 (Conv2D)	(None, 210, 270, 72)	93384
conv2d_15 (Conv2D)	(None, 210, 270, 48)	31152
batch_normalization_3 (Batch Normalization)	(None, 210, 270, 48)	192
up_sampling2d_1 (UpSampling2D)	(None, 420, 540, 48)	0
conv2d_16 (Conv2D)	(None, 420, 540, 1)	433
=====		
Total params: 437,617		
Trainable params: 437,233		
Non-trainable params: 384		

Model Summary by Keras



Model architecture generated using `plot_model()` in Keras

IV. Results

Model evaluation and validation

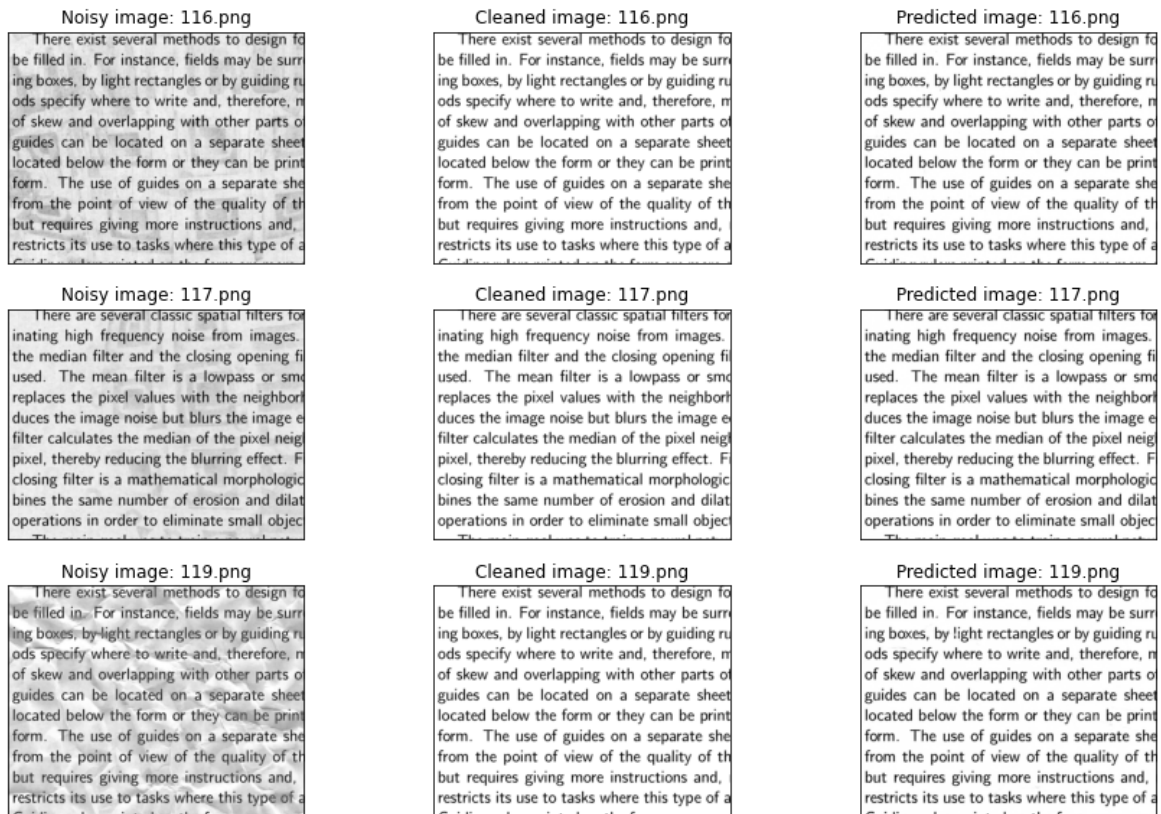
The autoencoder model achieved an RMSE score of 0.03948, along with an MSE loss value of 0.0012 and MAE value of 0.0154 on the training set after 168 epochs. The model was run for 200 epochs but there was an Early Stopping callback in place which stopped the training at 168 epochs since there was no improvement in loss for 10 epochs.

The Kaggle competition result is shown below:

Your most recent submission				
Name	Submitted	Wait time	Execution time	Score
submission3.csv	an hour ago	1 seconds	169 seconds	0.03948
Complete				
Jump to your position on the leaderboard				

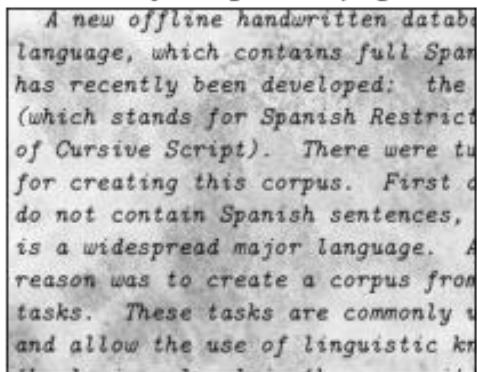
Kaggle Submission Score

A few samples of the training set, and the test set were passed as input to the model. The results are shown below:

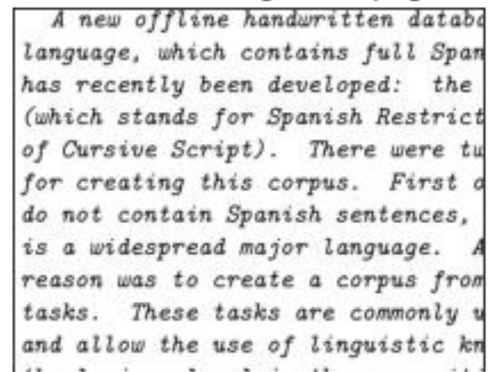


Results on a few samples of the training set

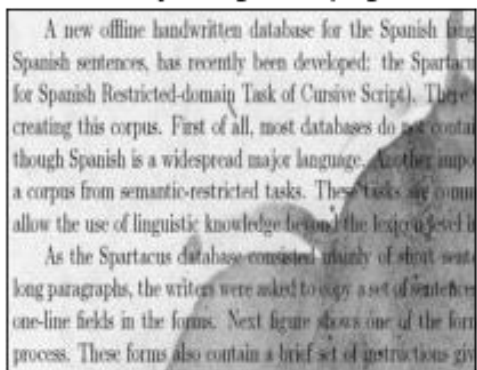
Noisy image: 127.png



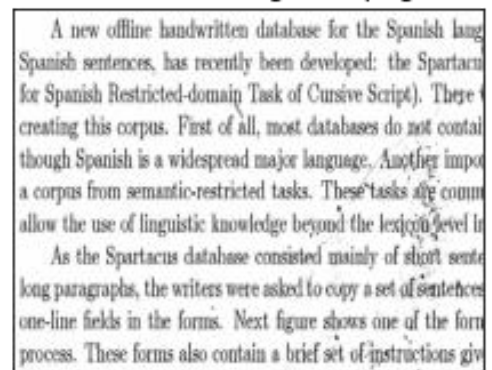
Predicted image: 127.png



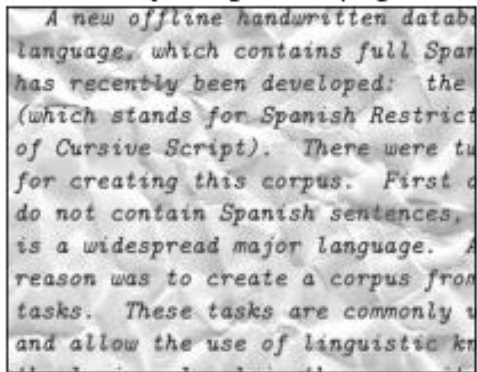
Noisy image: 13.png



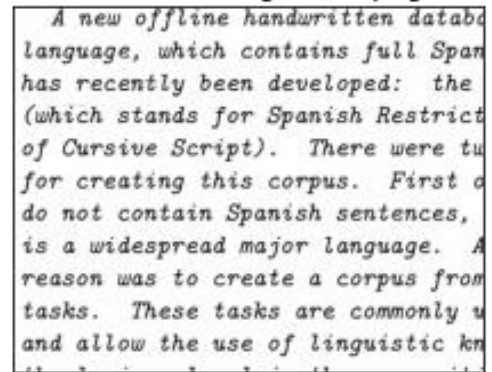
Predicted image: 13.png



Noisy image: 130.png



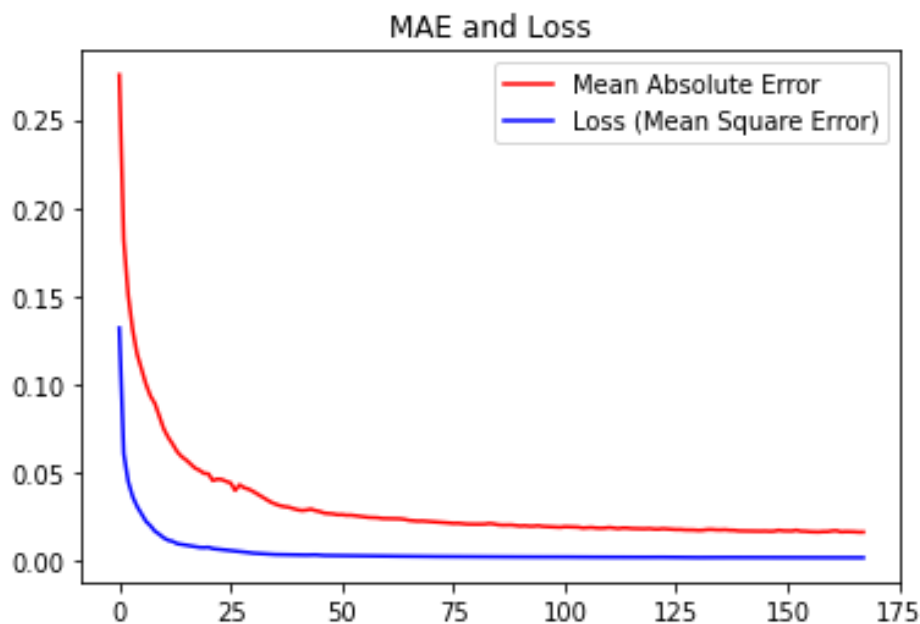
Predicted image: 130.png



Results on a few samples of the test set

Looking at the images above, we observe that the noise has been removed successfully, and that the output images are close to the training cleaned images as well. In some cases, there is still some noise remaining – specifically looking at ‘13.png’ – so all the noise has not been removed, the model has worked well for most of the cases otherwise.

Following is the plot of the MSE loss and MAE.



A plot of the MSE loss and MAE

The MAE steadily decreased till around 50 epochs, after which it continued a downward trend, though very slightly. The MSE loss however, quickly dropped down till 25 epochs after which it remained consistent around 0.0013.

Looking at the loss here, there may be overfitting in the model since the size of the training set is small, and it is unclear how well this model will generalize to new samples with different types, shapes and patterns of noise in the document images.

Justification & Conclusion

As mentioned in the results section above, the autoencoder model achieved an RMSE score of 0.03948. While not as good as the Kaggle top 10 leaderboard scores, this would place around # 62 in the frozen leaderboard. I consider this a good result, since no image augmentation or other similar techniques were applied. By applying such techniques, or by further fine-tuning the model architecture and hyperparameters, the score can definitely be improved. That said, further fine-tuning may also lead to overfitting (which may already have happened in the current model) due to a small amount of training data.

While considering the output images, the performance is adequate such that this can be used reliably for removal of the specific noise types as described in the Data Exploration section. The model is able to handle noisy documents with different types of noise which were not present in the training data – a good example of this is '119.png' in the figure in the previous section. Considering all these points, I believe this can be used with a wider set of documents with more variations in noise and can be effective especially when used in a pipeline to automate information extraction from scanned documents, provided that the model is trained on similar - if not the same - types of noisy documents that it would be expecting when running inference.

V. References

1. <https://blog.keras.io/building-autoencoders-in-keras.html>
2. <https://www.tensorflow.org/tutorials/generative/autoencoder>
3. <https://www.datacamp.com/community/tutorials/autoencoder-keras-tutorial>
4. <https://www.pyimagesearch.com/2020/02/17/autoencoders-with-keras-tensorflow-and-deep-learning/>
5. <https://www.pyimagesearch.com/2020/02/24/denoising-autoencoders-with-keras-tensorflow-and-deep-learning/>
6. https://en.wikipedia.org/wiki/Salt-and-pepper_noise