OCR text extraction in medical images

AKASHDEEP SINGH, University of Calgary, Canada

A planned model pipeline for filtering text embedded in medical images, and filling replaced pixels to restore the look of original image, minimizing leakage of sensitive user information.
The pipeline involves multiple stages as follows, text classification, text extraction and, finally, image painting which is expected to allow a modular but robust technique to separate the embedded text from the image without leaving any clues on the painted image.
In this project, I was able to achieve the 1st checkpoint, i.e. the first stage of making a classifier of the type of embedded text an image contains, whether it is printed and/or handwritten. The method of the classifier was approached using a resnet18 with the modification of final **fc** layer to a dropout and linear layer, allowing to achieve with a resulting accuracy of 98%.
While printed text has existing sophisticated models to extract it, the same does not apply to handwritten text, as it imposes greater challenges.
Reaching the the first milestone has been a great success in achieving it, and future works will proceed onto exploring and researching in the next stages. A complete architecture of the model and discussion is provided in this report.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Machine learning approaches**.

Additional Key Words and Phrases: handwritten text recognition, image painting

ACM Reference Format:
Akashdeep Singh. 2018. OCR text extraction in medical images. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

With over millions of patients medical scans performed, significant and sensitive data needs to be protected from the working staff, without compromising the evaluations of the reports.
Many medical images contain embedded text, which might identify patient's data, and separating the two independently is often an uneasy task. One approach is although manual, where human intervention allows to filter text data and refill the image, however that exposes the data to the human in action itself. Rather an automated computation is a better solution and alternative which not only has the potential to anonymously complete the task but also process it at greater speed and efficiency.
Although, the challenge lies in ensuring both the quality of the medical image and minimizing the leakage of any sensitive data embedded.
However, text may come in mainly two forms:

- printed: follows a rigid vectorized and mathemticaly shape of each letter, which allows a more sophisticated character recognition and extraction. And in fact, existing solutions are already above satisfactory performance.
- handwritten: it is unique for each individual and does not have a specific pattern therefore. Often, letters or pair of letters are written in different shapes and curvatures which implies a larger complexity to recognize them. Existing solutions are present, but although still require to discover more potential in greater effectiveness and accuracies.

In this paper, we discuss a project of a planned model pipeline that allows extract both types of texts and replace the text with painted image that resembles as closely as possible to the original image.
The pipeline does not only apply to medical images, but to even a broader context. In many applications of computer vision, text extraction is a demanding and popularly becoming feature that has emerged in most common day application. Moreover, handwritten text extraction demands newer and more sophisticated model to achieve better statistics.
tess

2 MODEL ARCHITECTURE

2.1 Pipeline

The architecture of the model is the following pipeline in the figure shown (Fig. 1).
First, the classifier will determine which type of text the image

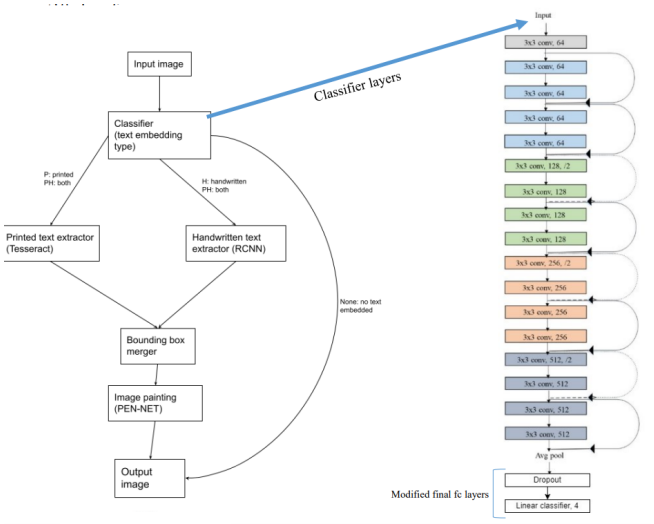


Fig. 1. Model architecture and pipeline.
On the right side, the detailed structure of implemented classifier model is shown.

contains. Then, if any text embeddings are present, based on that the image will be processed by printed text extractor and/or by the

Author's address: Akashdeep Singh, akashdeep.singh4@ucalgary.ca, University of Calgary, Calgary, Alberta, Canada.

0730-0301/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

handwritten text extractor. All the extractors are expected to return the text found and as well the bounding boxes.
Finally the bounding boxes of the image are replaced and painted by the final layer, which applies image painting. The re-painted image should resemble as close as possible to the actual original image without any text embeddings.
Therefore, the pipeline involves training each layer independently and hence why the custom dataset should support different label output modes.

2.2 Justification and Possible improvements

The classification model is designed with **resnet18** layer, with the final layer replaced to 4 output classifier (None, P, H, PH). The reason for choosing such a model is since that in order to allow proper feature extraction and then classification of the image, the model needs to be enough deep and complex, but at the same time efficient enough to be effective in practical use. We first started with resnet18 as our first choice, since 18 layers can provide enough depth for model complexity but at the same time effective in practical use to allow lower computation usage and delay. Initially, the model was planned to involve transfer learning technique by training only the last 4 output classifier. However, due to low accuracies the classifer was improved by training the whole model layers.
Since printed text is easier to extract and has existing models already highly sophisticated, **tesseract** model is used for extracting text. Tesseract is a widely popular OCR model, designed to extract printed text from images.
However, extracting handwritten text is a bigger challenge and indeed the main-core of the project. And for exactly this reason, the pipeline was designed in this manner for potentially achieving better accuracies. The current plan, is to use RCNN layers to extract handwritten text, however an appropriate loss function is as well required to ensure proper alignment is done. Currently, the expectation is to use CTC loss function which seems best suited for such application. [Chaudhary and Bali 2022], [Memon et al. 2020].
Then, just before replacing bounding boxes with painted images, the penultimate stage of the pipeline is to ensure all bounding boxes are unique and conflicting ones are merged appropriately. This part does not involve any model training but rather designed to remove any conflicts.
Finally, the image requires re-painting and PEN-NET model is expected to complete the appropriate task. This existing model has achieved relatively high accuracies and efficacy in resembling the output image as closely as possible to the original image. Since the custom dataset, as well keeps track of the original images, the original image can as well be returned to allow the model to measure the appropriate the loss, when trained.
While, PEN-NET can be used as a pre-trained model, it is best to re-train as the dataset is specifically a medical image and a subset of the set of broad range of images. A medical image involves gray scale image and therefore, in order to achieve better accuracies it might be better to re-train the model.

3 DATASET

3.1 Custom Dataset

As there is not publicly available medical image dataset, which contains embedded text and corresponding labels, a custom dataset had to be made using an automated script process.
First, a publicly available dataset of COVID ct-scans was extracted, which contains original images without any text embedded inside it. Then, for each of these original images, text has been embedded insider them, both printed and handwritten. The embeddings were done by merging text image(s) inside the original one. The background of text images was treated transparently to allow a good level of embedding. If the background of text image was opaque in the embedded result, then the model would falsely behave too accurately as it is easy to find such regions.
For generating printed text images, random sentences strings were converted to images of different sizes in a simple fashion.
Whereas handwritten printed text images were pulled from existing IAMS dataset, as unique handwriting styles are simply too many.
Thus, in simple words the custom dataset was generated as follows:

(1) Generate text dataset of printed images
(2) Generate text dataset of handwritten images
(3) For each medical image, generate multiple images with various text embeddings
    - None: no embeddings
    - P: printed embedding(s)
    - H: handwritten embedding(s)
    - PH: both

The size of the custom dataset is approximately **4500 images**, involving all of 4 types of text embeddings (making it **diverse**) and the source was from SarsCOV2-CTSCAN-dataset from kaggle, as previously mentioned.

3.2 Processing and Augmentation

Each image from the custom dataset was preprocessed to a rescale of fixed resolution (500 x 500), followed by augmentation. On a basis of probability, color jitter and grayscale transformation were applied.

3.3 Biases and Limitations

The custom dataset does not contain text embeddings with rotations or other transformation such as shear or curvatures. This is a limitation as the trained model pipeline, if successful, would not be able to extract text accurately from images with such complex embeddings. However, this project's focus is to generate a more viable approach by beginning with a limited complexity level. The more complex the dataset becomes the more difficult it might be to first assess the model.
This is also the reason the augmentations on the dataset did not involve rotations and any other transformations, as the model might potentially need to be more complex to achieve the same level of accuracies.

The goals of the project are to assess if the approach is viable, before adding more enhancements to the pipeline to support a broader

range of dataset. Thus, future works may improve and upgrade the pipeline for the latter aspect of the project.

Moreover, the model pipeline is designed to extract text from medical images, specifically. While a more general model may be more beneficial for extracting text from any type of image, the accuracy may be suffered.
Therefore, focusing and assuming that the dataset consists only of medical images, where the majority is in grayscale color, the data augmentations were also exactly applied for that scope. This will also allow the model to be trained more well to dataset with specific type of images, so that accuracies are higher.

Another note is that the embedded text is always black. While it is true the model could be trained with a broader range of text colors, the custom dataset was simplified to the fundamentals for the same reason as above.

In summary, the dataset is indeed a subset of wider and broader range, although still sufficient to generate a model. Future works and enhancements, can later lead to more additions.

4 TRAINING

PyTorch is the framework used under Google Colab with GPU runtime enabled.
The model used in resnet18, as pretrained, with the final layer modified to 4 output classifier. Initially, transfer learning was applied by only allowing to optimize and change weight of the final layer. However, the accuracies were not sufficient and around 60%, implying the model was too underfitting.
The initial batch size of dataloader was also low (batch_size = 4) which caused skewness in accuracies between epochs. This is because after each batch size the model would update the weights and happened too frequently (so the momentum was not very effective). Thus the first improvement was increasing the batch size to 32, which allowed the accuracy between epochs to be more stabler and instead of sudden jumps it was continuously improving and increasing.
However, as mentioned the model was still underfitting due to only training the final layer and thus fewer amount of weight parameters. Thus, all layers of resnet18 were unfrozen and thus allowing to optimize more weights. Moreover to **prevent overfitting** in the model, dropout layer was prepended before the classifier, so that the model generalizes well enough. Without it, the model was overfitting by achieving training accuracies above 99% and validation accuracies below 85%, with improved/reduced loss on training and worsened/increased loss on validation dataset. It is true that using Drouput there is risk of killing the gradient in optimization of weights using gradient calculation, however the Dropout layer is in the final stage of the resnet18 structure and therefore as seen in the results of accuracy it instead helped to not overfit and increase accuracy, instead of the opposite.
The optimizer used was SGD with a learning rate of 0.01 and momentum of 0.90. Cross entropy was used to measure the loss function.
The results of the classifier with this improvent indeed allowed to achieve accuracies up to 98% after 20 epochs. Moreover, it has been noticed that increasing number of epochs beyond 20 (such as 50) did not improve the accuracy and rather the validation accuracy started to drop due to possible increased overfitting.
Thus, the model has been able to get trained within roughly 20minutes with a great level of accuracy. You may also see Fig. 2 to see the statistics.

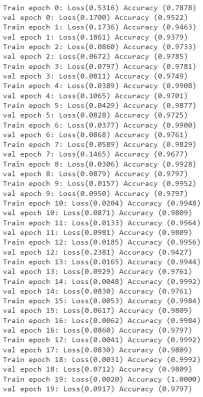


Fig. 2. Model training statistics.

Regarding the loss functions (see Fig. 3) after roughly 10 epochs a bump of the validation loss is quite noticeable. This likely implies that after that point the training model may potentially start overfitting, however after that point it has gone to a lower stabler loss value. In fact, even after it is stabler the training loss curve is slightly decreasing whereas the validation loss does not, after that spike. Thus, indeed after looking at the stats in Fig. 2 in the 10th epoch the accuracy is already at 98%. This implies that Early Stopping may have been done at 10 epochs, although the model has been kept training till 20, as at the end the validation accuracy did not worsen overall and the model did not significantly overfit, thanks to the use of Dropout layer.

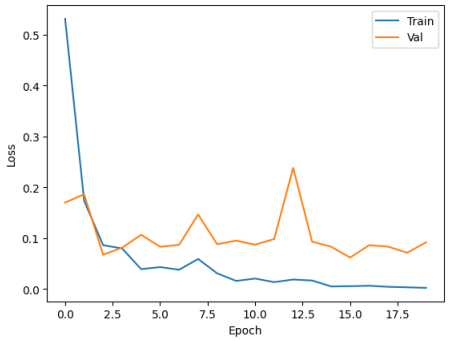


Fig. 3. Model training curve loss statistics (training and validation loss) in 20 epochs.

5 EVALUATION

5.1 Metrics

The accuracy was the evaluation metric used to assess model's performance. Moreover, at the final stage a test dataset was also tested to ensure accuracy of model is consistent with both validation and test dataset. Fig. 4 shows the consistent accuracy of 98% just as seen previously in the final epoch of validation dataset.

```
In [26]: evaluate(model, loader, device, criterion, mode='test')
test epoch 19: Loss(0.0917) Accuracy (0.9857)
Out[26]: 0.09172217964807632
```

Fig. 4. Model evaluation with test dataset

Deeper models such as resnet50 were also tested however, the overall accuracy did not exceed 98% as quality of dataset had less noise. The benchmark on Fig. 5 shows a quantitative comparison of the different resnet models that can be used. As it can be seen, when the amount of noise is relatively very low then resnet18 performs as good as any of other deeper resnet models.

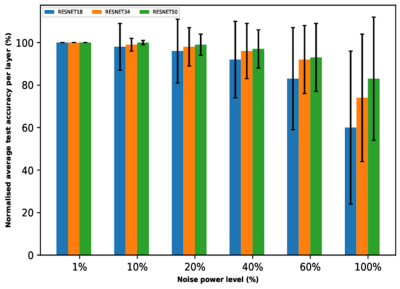
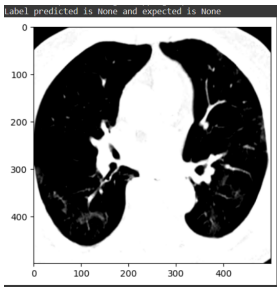


Fig. 5. Benchmark comparison between different resnet layer depth models.

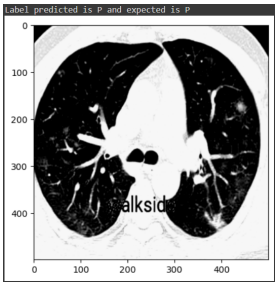
5.2 Qualitative samples

In Fig. 6 a qualitative sample is provided for each of the possible outputs of the classifier, provided sample input images from the test dataset.

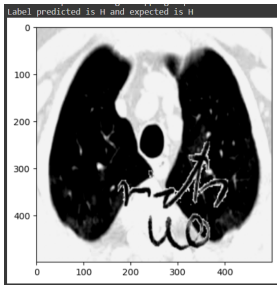
- Fig. 6a shows image with no text embedding at all. The classifier in this case should inform the model pipeline to directly pass the input image as the output since no image text extraction and painting is required. Notice the predicted and expected label **None**.
- Fig. 6b shows image with printed text embedding. The classifier in this case should inform the model pipeline to directly pass the input image to the "printed text extractor model". Notice the predicted and expected label **P**.
- Fig. 6c shows image with handwritten text embedding. The classifier in this case should inform the model pipeline to directly pass the input image to the "handwritten text extractor model". Notice the predicted and expected label **H**.
- Fig. 6d shows image with both types of text embeddings. The classifier in this case should inform the model pipeline to directly pass the input image to both "printed" and "handwritten" text extractors, which will undergo a bounding box merger to ensure no conflicts are present before the final stage of image painting. Notice the predicted and expected label **PH**.



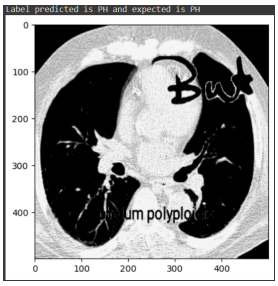
(a) Sample image with no text embedded.



(b) Sample image with printed text embedded.



(c) Sample image with handwritten text embedded.



(d) Sample image with both text types embedded.

Fig. 6. Sample images with different text embeddings.

Moreover, notice how in all these cases the predicted label matches to the expected ones for that fact the accuracy is high. It is hard to find counterexamples, other than building an iterative script, however the purpose was to show how each of the outputs of classifier would look like for each type of image.

5.3 Model limitations and future works

Regarding the classifier again, the dataset with text embedding may possibly cause model to sometimes predict wrong value. The main reason is when very little characters are present in the embedded image, which is hard for the model and human itself to trace. For example a comma, punctuation signs may cause sometimes model to predict as no text embedddings are present. However, one possible improvement to allow model to achieve better accuracies is to remove from the dataset such image embeddings with minimal text or characters.

The first checkpoint has been reached in the model pipeline and future works involve adding handwritten and printed model, followed by image painting. As previosly mentioned, the classifier and all model stages assume text embedded is rectangular and in a bounding box, without any other possible shapes such as circles, curves etc.. The main priority is to create and assess a mode for simpler subset of dataset and then over time in the future newer features can be addded to allow model to support a more complex set of dataset.

6 DISCUSSION

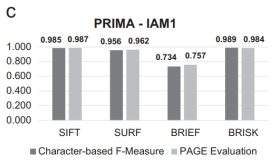
6.1 Results

Comparing to existing work of a classifier [Garlapati and Chalamala 2017], [Zagoris et al. 2014] the accuracy indeed matches around 97-98% (using the IAMS dataset) as shown in Fig. 7. This is almost as close with a marginal difference of the results and within a few epochs and short amount of time. Therefore, the project first stage of the pipeline shows a great level of success.

TABLE II
CLASSIFICATION ACCURACY OF HANDWRITTEN AND PRINTED WORD SEPARATION FOR IAM DATASET

S.No.	Cross fold Numbers	Classification Efficiency
1	Cross fold 1	97.91
2	Cross fold 2	99.84
3	Cross fold 3	99.86
4	Cross fold 4	98.42
5	Cross fold 5	99.12
6	Cross fold 6	97.23
7	Cross fold 7	98.53
8	Cross fold 8	99.78
9	Cross fold 9	98.25
10	Cross fold 10	97.78
11	Total Average	98.6

(a) Classifer results of [Garlapati and Chalamala 2017]



(b) Classifier results of [Zagoris et al. 2014]

Fig. 7. Classifier results of other works. Both papers show a similar accuracy in the range of 97-99%, each with their own setup of classification model architecture.

The implications of this high accuracy rate in the context of computer vision are:

- **Increased Efficiency**: The high accuracy rate of the classifier means that you can save time and resources that would have been required for manual sorting of printed and handwritten images. This can be especially valuable in medical contexts where time is of the essence.
- **Improved Data Quality**: By accurately identifying images with printed or handwritten text, the classifier can help ensure that subsequent analyses are based on high-quality data. This can lead to more accurate and reliable results.
- **Potential for Automation**: The high accuracy rate of the classifier also suggests that there may be potential for further automation of the OCR pipeline. For example, if the pipeline is able to accurately classify images as printed or handwritten, it may be possible to automatically select the appropriate OCR algorithm for each image.
- **Future Development**: If the pipeline's high accuracy rate is successful then it may also open up opportunities for further development and improvement. For example, if the pipeline is consistently able to accurately classify images as printed or handwritten, it may be possible to train the classifier to recognize other characteristics of the text (such as language, font, or size). These are exactly the successive steps in building a better pipeline on top of the current one.

6.2 Further Research or Applications

The project's model pipeline is a new and unique experiment in progress. The classifier model of first stage of model pipeline shows that handwritten and printed text can be effectively distinguished so that it becomes easier in the latter phase to extract the 2 types of text independently. Training a model for handwritten and printed text work best if done separately as printed text font is consistent and has different patterns then handwritten one.

Similarly handwritten text cannot be categorized as a whole single type only. Each user's handwriting style is different and thus different patterns exist. Further research in determining different types of handwritten text can allow to build a more complex but effective model pipeline for handwritten text extraction. For instance, once handwritten text is classified, another classifier can be used to determine the specific type of handwritten text so that it text is extracted by a specific model.

One concern, might be the conflicting cases where at least 2 models end up extracting the same or some portion of same text. In that case, the conflicts should be still handled by the bounding box merger pipeline which serves for that purpose. Another related concern is if the conflicts are same bounding box but result in different text. This indeed seems to be a major limitation and potential issue for the proposed model pipeline, as the bounding box merger cannot distinguish between which of the 2 text is correct. One proposed solution is to indeed re-use the classifier **within** the image delimited by the bounding box to determine the most likely correct text extraction.

Therefore, in other words, the model pipeline is expected to be a robust and sophisticated technique for text extraction of various types of text, with the potential of further improvements. Although the main work lies in the future which will allow to determine whether the model pipeline has potential in a better OCR or not, in the context of computer vision.

7 CONCLUSION

In summary, the proposed project shows a model pipeline to extract both handwritten and printed text on an image, with a specific focus on text embedded in medical images.

The first checkpoint achieved in the project shows a robust and effective classifier that determines the type of text embedding present in the image. Future works include the next checkpoint of the model pipeline where the image's printed and handwritten text will be extracted followed by image painting.

The work lies on generating a model for handwritten text extraction, as existing models are already sophisticated for printed text extraction and for image painting. Whereas for handwritten text extraction the main work lies in the future.

REFERENCES

Kartik Chaudhary and Raghav Bali. 2022. Easter2.0: Improving convolutional models for handwritten text recognition. arXiv:2205.14879 [cs.CV]

Bala Mallikarjunarao Garlapati and Srinivasa Rao Chalamala. 2017. A System for Handwritten and Printed Text Classification. In *2017 UKSim-AMSS 19th International Conference on Computer Modelling & Simulation (UKSim)*. 50–54. https://doi.org/10.1109/UKSim.2017.37

Jamshed Memon, Maira Sami, Rizwan Ahmed Khan, and Mueen Uddin. 2020. Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR). *IEEE Access* 8 (2020), 142642–142668. https://doi.org/10.1109/ACCESS.2020.3012542

Konstantinos Zagoris, Ioannis Pratikakis, Apostolos Antonacopoulos, Basilis Gatos, and Nikos Papamarkos. 2014. Distinction between handwritten and machine-printed text based on the bag of visual words model. *Pattern Recognition* 47 (03 2014), 1051–1062. https://doi.org/10.1016/j.patcog.2013.09.005