Manga Bubble Segmentation

By Charles Chan, Sally Bao

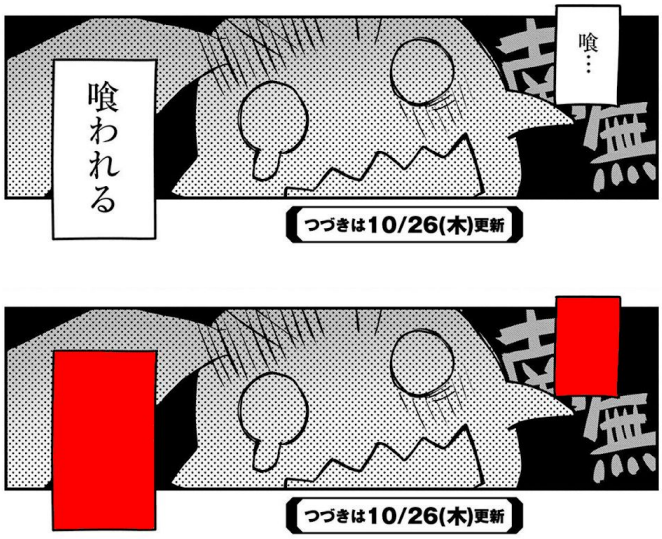
Code and docs

Overview

The purpose of this project is to perform image segmentation on comic and manga pages in order to identify and isolate speech, text, and thought bubbles. We hope to use the results to expedite the process of manga localization, one step of which is to clean out the original text in all bubbles so that translated text may be set over it. The two approaches for the project were to identify whether a given region has a bubble, and to segment the bubble directly from the image, producing a mask.

Data

Our initial data was obtained from an untranslated (raw) manga website as grayscale images of each page. We manually labelled the regions that contained bubbles with colors to create a mask as well as identify whether a bubble is detected. We created a black and white mask of bubbles with the labelled image and used a sliding window approach to cut each page into smaller segments, which we separated into two classes: contains a bubble or doesn't. We assumed a threshold of 70% of a bubble's contours indicates a bubble is present.



Due to the time consuming nature of manually labelling the data, we only produced 229 labelled images. To increase the dataset, we performed various transformations such as scaling,

rotations and mirroring before using the sliding window to cut the image. This resulted in a total of 2.6 million 256 by 256 pixel images. This seemed reasonable as the varying scale and orientation should not affect whether a human can identify a text bubble in an image.

A separate data transformation we did on the pages was to use an existing OCR library to detect Japanese text on the manga and produce black and white masks of text regions. We decided to use this set of data for last as there was a noticeable amount of false positives visible.

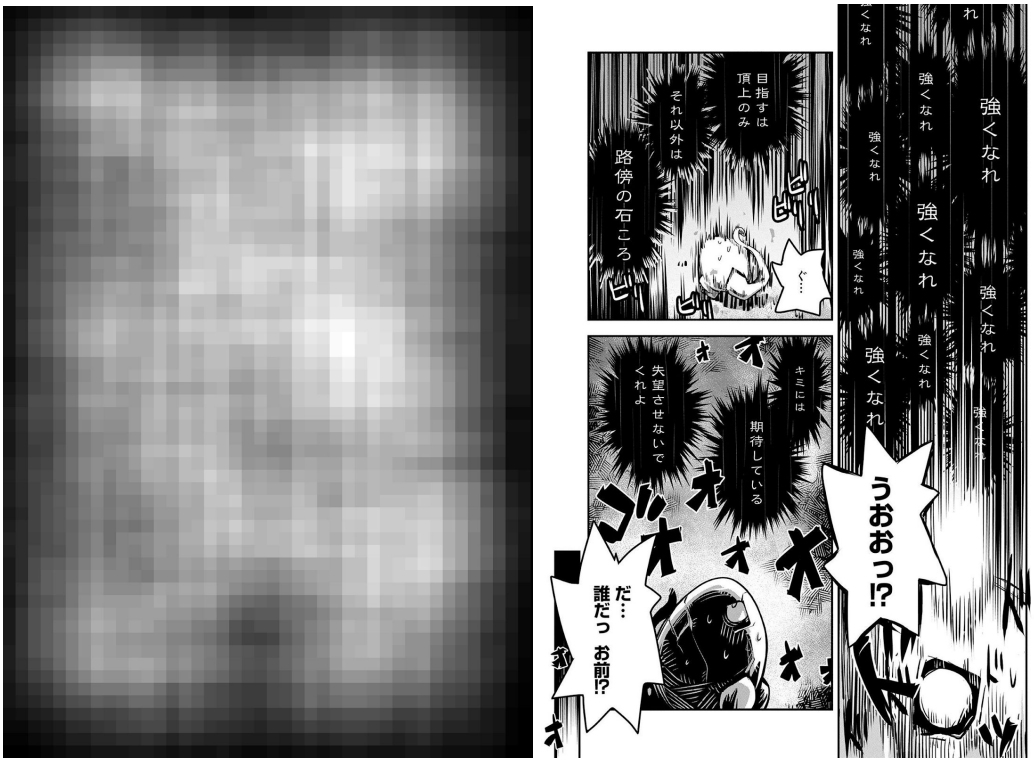
OCR detector falsely captures the figure as text.

Models

Due to time and resource constraints on this project, we were not able to explore more complex model training optimizations like regularization and dropouts. Our evaluations were limited to a validation set produced from the samples and manually comparing the heatmap output to the original image.

Logistic Regression

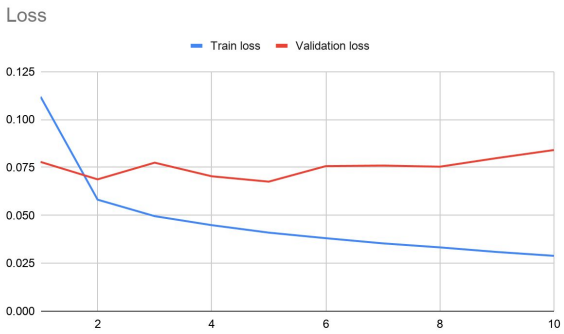
The model used for a baseline was a simple logistic regression as it was a fast and simple binary classifier. Due to memory limitations, the model was trained on a random sample of 20,000 images. It achieved around 60% accuracy each time on a random sample of 20,000 validation images, slightly higher than our predicted baseline of 50%.

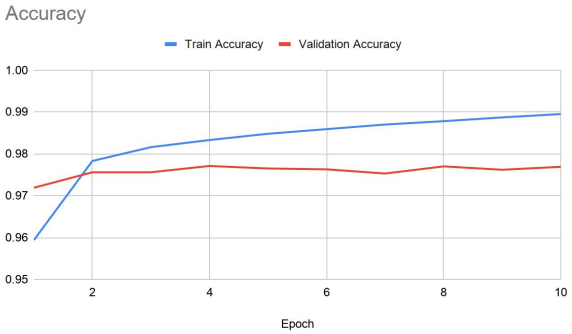


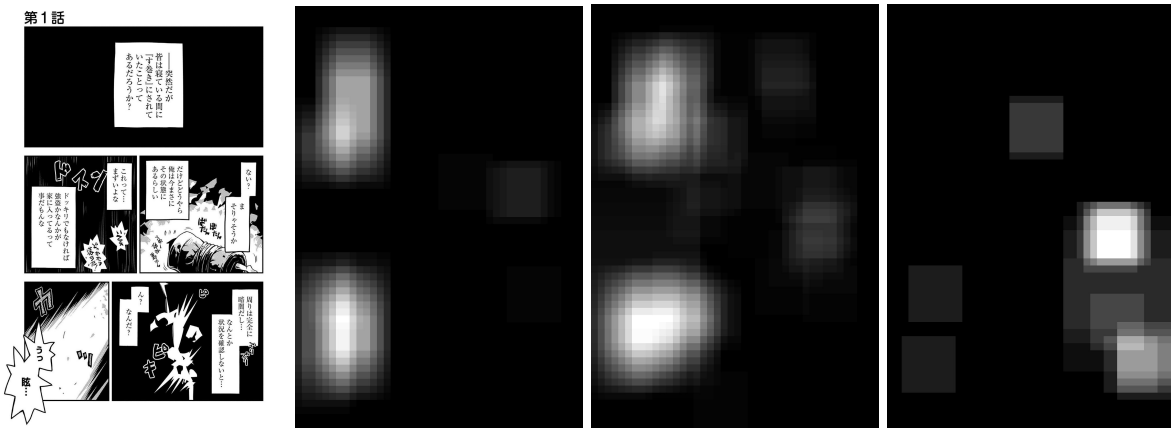
Heatmap produced by running a sliding window over a page and detecting bubbles using the logical regression classifier.

Convolutional Neural Network

Since we were using image data, we thought it made sense to try a convolutional neural network classifier next, as the convolutional layers could capture features like edges and make it easier to detect bubbles. As it was using a neural network, it was possible to use the entire dataset. The model was trained with a set of 100k bubble images and 1.5 million non bubble images for a baseline accuracy of 93%. The remaining million images were stored for validation. While training results steadily improved with each epoch, there was little to no improvement on the validation data.



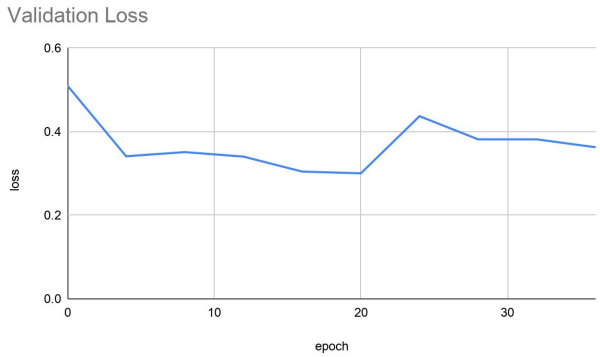


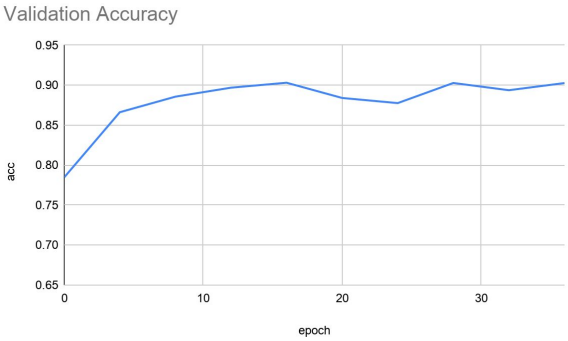


Heatmap produced with CNN binary classifier. Shows significant improvement over the logical regression classifier.

Convolutional Autoencoder

In order to segment the bubbles in an image, we started with a simple autoencoder, mapping images to their desired output masks. Due to memory limitations of loading both images and their masks, the model was trained on chunks of the data for a total of 102K bubble images and 204K non bubble images, with a baseline accuracy of about 86%. A train-test split of 3:1 was used. The validation accuracy improves slowly over the epochs, while loss sees a general decrease.





Running the model on a random image, with true mask (center) and predicted mask (right).

Convolutional Neural Network Autoencoder with Skip Connections

In an attempt to improve the quality of the initial autoencoder, we designed a model based on an article describing convolutional autoencoders. In particular, the concept of using skip connections to bring the previous output of the encoding component to the decoding component may be useful in sharpening the edges of images and filling the interior of the mask more fully. In between the convolutional encoder and decoders, a fully connected network was inserted to detect more complex features.

Clearer predicted mask with CNN autoencoder.

OCR Autoencoder

Another idea we tested was incorporating the output of the OCR library into the training. This model is an autoencoder with a similar structure to the first one, but takes two inputs, the original grayscale image and a mask produced from the segmented text regions. We thought the text mask could help the model identify regions with bubbles, while the network could cut down the number of false positives produced by the text detector.

Conclusion

For the binary classifiers, we did see a large improvement from the logistical regression model to the convolutional neural network. This was expected as the convolutional network had more layers to detect features and more data to train on. Because we didn't have enough time to fully train the later autoencoders, it is hard to compare their performance with the simplest autoencoder. We expected the later approaches to have better results if given the time to train, since they were composites of our simpler models.

There were many things we could have tried but due to constraints on our time and resources, as well as mistakes along the way, we were unable to. While we did get some results from our models, they are not satisfactory. Given more time, we could've labelled more pages of data from different mangas, adding to the variety of data available and making our models more robust. We could've considered other augmentations like inverting the grayscale colors and adding random noise to the samples. We could've tried improving the performance of reading from the disk by merging multiple image samples into a single large image to be parsed in the virtual machine's memory. We were also planning to try combining our heatmap approach with the autoencoder: run a sliding window over a page to determine bubble regions with the classifier, then feed that input into the autoencoder.

A promising approach we could've looked at was generative adversarial networks. Since small differences in a bubble's shape are insignificant when determining whether it is a bubble, this approach may be effective despite our lack of variety in the data.

Due to the variety of different shapes and patterns used in manga bubbles, one thing we could consider is whether it would be more effective to train models that are more specialized towards each bubble type (information bubbles, speech bubbles, thought bubbles) as well as common bubble shapes. However, because there are no definite rules about what a bubble is supposed to look like, this task is in itself difficult and may limit the accuracy if the model becomes too specialized.

Overall this project was not something we could've completed in a short span of time with limited resources. We'd like to think of this as an initial exploration of the dataset and potential solutions. These initial results seem promising and will probably improve given more time.