MangaNet: Building an Object Detection System for Mangas

Tai Vu
Computer Science
Stanford University
taivu@stanford.edu

Robert Yang
Computer Science
Stanford University
bobyang9@stanford.edu

Abstract

Object detection in mangas is a difficult problem due to the large amounts of whitespace, the grayscale format, large amounts of overlapping objects, and the large number of variants in shapes of objects. In this project, we use a number of object detection models (FasterR-CNN, RetinaNet, and YOLOv3) to localize and classify objects in mangas as face, body, frame, or text. We also experiment with techniques like transfer learning and data augmentation, and test out a featurizer based on a variant of Neural Style Transfer. We find that YOLOv3 achieves the highest performance with a mAP of 71.0 on the Manga109 dataset.

1. Introduction

In recent decades, the production of mangas has become a large commercial industry that receives great public attention. In addition to offering fascinating plots and character development, manga series are a means of popularizing traditional and cultural values. The exponential growth of digital technologies like iPhone, iPad, and Kindle has facilitated the distribution of comic products across the world. Hence, building computer programs that can understand manga contents has become an increasingly important task.

The recent advent of machine learning has driven major advances in computer vision systems, which provide great tools for automatic analyses of manga products [8]. In this project, we will focus on the problem of object detection in comic series using deep learning approaches.

We will develop convolutional neural networks (CNNs) to detect the presence of certain elements in manga pages. In particular, our models will take as inputs images from manga series and outputs bounding boxes for all detected objects and their corresponding labels. In this study, we evaluate our algorithms on the Manga109 dataset [16, 19] and consider four bounding box classes: "frame", "text", "body", and "face".

There are four main difficulties of the Manga109 dataset as compared to other object detection datasets. First, there is a large amount of whitespace in manga sketches. This makes it difficult for object detectors to latch on to textures that may be useful for detecting objects, and outweighs the seeming cleanness of the images. Second, with the exception of some title pages, most manga pages are drawn in grayscale due to the high time cost of coloring. This means that there is less information for the object detector to use, and the object detector may not recognize objects that they might have recognized if it was in color. Finally, mangas have many overlapping objects, so it is very difficult to detect the presence of each object individually. Finally, compared to real-life photos, there are much more variants when it comes to the shapes, sizes, forms and styles of manga characters' faces and bodies. These patterns pose a great challenge to our models when detecting the presence of certain objects in comic pages.

We will train three different models (FasterR-CNN, RetinaNet, and YOLOv3) with variations (including transfer learning and data augmentation) and compare the results. We will also tailor our pipeline specifically for manga comics by experimenting with Neural Style Transfer as a featurizer for object detection in mangas.

2. Related Work

With the availability of massive data sets and the accessibility of computing reesources, there have been a large amount of research on deep learning based object detection algorithms. Some common approaches include R-CNN [6], Fast R-CNN [5], and Faster R-CNN [25]. These models extracted region proposals for each object in the input images, and then feed them into a CNN network to produce output labels. Another popular object localization algorithm is YOLO [22], which joinly predicted bounding box coordinates and class probabilities. Recently, a new CNN-based technique called RetinaNet was proposed [10], which leveraged focal loss and achieved state-of-the-art outcomes in object detection benchmark datasets.

Over the last few years, there have been a number of studies on the application of machine learning and computer

vision algorithms in comic domains. For instance, Chu et al. leveraged several features like speech balloons, shapes of frames, and numbers of characters to analyze comic styles [2, 3]. Meawhile, Arai & Tolle utilized graph-based connected component detection algorithms on white and black pixels to detect panels and balloons [1]. Similarly, Rigaud et al. combined connected component labeling with k-means clustering to improve the classification of texts and frames [26]. These techniques work well with simple manga pages, but fails when there are non-rectangular frames or complex sets of objects. Other downsides of these approaches are their slow runtime and relatively low performance, so they are not practical for production use cases.

Other algorithms required several classical feature engineering techniques to detect certain objects in comic pages. For example, Sun et al. combined SIFT descriptors and local feature matching to tackle the problem of character detection and identification [28]. Likewise, Matsui et al. harnessed EOH features and approximate nearest neighbor search to localize characters' faces and bodies [16]. While these techniques are simple and do not require large training datasets, they are quite computationally inefficient and do not generalize well to diverse sets of manga styles.

In addition, some recent research projects have focused on the use of deep neural networks for comic analysis. Nguyen et al. developed a YOLOv2 region proposal network to detect bounding boxes around frames and characters, which achieved a slightly better outcome than traditional feature engineering methods [18]. However, their model focused more on color manga datasets and neglected its application in black-and-white comics, which is the most popular form of comic in the current market. Likewise, Ogawa et al. experimented with several deep learning methods like SSD and YOLO for object localization in comic pages [19]. Nonetheless, the use of deep learning for in-depth analyses of manga products is a rather underexplored domain. This motivated us to build CNN models to tackle the task of object detection in manga images.

3. Dataset and Features

3.1. Data

We are using the Manga109 dataset [16, 19], which consists of $10,619$ pages (images) of manga from 109 different Japanese books. Each image has a resolution of 1170 by 1654. Of these $10,619$ images, $10,130$ have a nonzero amount of annotations. It is this condensed dataset of $10,130$ that we are using. The dataset has four classes of objects: "frame", "text", "body", and "face".

The annotations for each image are in the XML format, which we later converted to CSV for more ease of processing. Meanwhile, the dataset is split into 80% for training, 10% for validation, and 10% for testing.

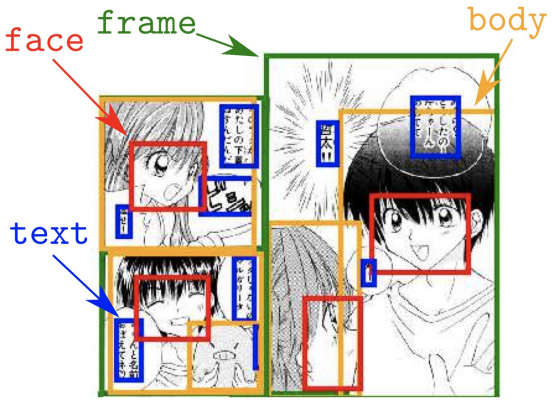

Figure 1. An example in the Manga109 dataset [19]

3.2. Data Augmentation

We applied data augmentation techniques in order to prevent overfitting and improved generalization power. These techniques include randomly removing a fraction of pixels, random affine transformations (translation, rotation, scaling), changing brightness, and horizontal flipping.

3.3. Neural Style Transfer

Four difficulties of the Manga109 dataset (whitespace, grayscale images, overlapping images, and a wide variety of styles) were mentioned in the introduction. As a potential solution to the first and second difficulties, we experimented with using Neural Style Transfer as a featurizer. There has been previous research in object recognition for objects rendered in artistic modalities (cartoons and sketches) that suggests style transfer as a possible method to increase the accuracy of object detectors on sketches [30]. However, they focused on generating training data (i.e. augmentation during training). In contrast, we directly run all images in the dataset through the MSGNet neural style transfer as a preprocessing step to create a modified dataset identical in structure to the original. This might be a solution to the first and second difficulties of the Manga109 dataset. First, any whitespace will be filled with the style that was transferred. We theorized that this will make it easier for object detectors to latch on to certain parts of the image. Second, style transfer inherently comes with some minor colorization, which may make it easier for the detector to see objects' differences. One of the images that we chose as the style image is shown in Figure 2.

4. Methods

Our main modeling approaches include Faster R-CNN [25], RetinaNet [10], and YOLOv3 [24].

Figure 2. A style image.

4.1. Faster R-CNN

Faster R-CNN [25] is an end-to-end network with multiple modules: RPN (region proposal network) and Fast R-CNN [5] (refining the bounding boxes and classification). The two modules share most of the convolutional layers to increase efficiency and performance.

The input to Faster-RCNN is a flexible size up to 1000×600 pixels, which is fed into the convolutional stem (feature extractor) that feeds into both the RPN and the Fast R-CNN modules.

The Region Proposal Network (RPN) [25] took inspiration from the concept of attention – it is responsible for telling the Fast R-CNN where in the image to look. This particular RPN of Faster R-CNN is different from previous papers as it is a small fully convolutional network. It takes a feature map as input from one of the later layers of the feature extractor stem. Then, a 3×3 convolution is applied, followed by two branches of 1×1 convolution. The first branch is for "box classification", which outputs the probabilities that the box contains an object or not. The second branch is for "box regression", which finds the approximate boundaries of any object in the box.

The Fast R-CNN module then takes as input the classification and regression results from the RPN as well as the same feature map that was inputted to the RPN. It is responsible for further refining the bounding boxes from the approximate boundaries that were generated by the RPN and classifying objects into different classes. After a filtering process to choose the boxes with the most confidence, the resulting bounding boxes and classes are then outputted by the network as the final output.

Faster R-CNN is trained with a joint loss calculated from the RPN outputs (object likelihoods and approximate bounding boxes) and the Fast R-CNN outputs (object class likelihoods and exact bounding boxes). The RPN loss component is calculated using random samples of 128 positive and 128 negative boxes to avoid bias towards negative samples.

In the original Faster R-CNN paper, the feature extractor used was VGG-16 [27]. In our study, we used ResNet-50 instead [7].

4.2. RetinaNet

RetinaNet [10] is a response to the major issue of imbalance that has hampered the performance of models such as YOLO and SSD. Its major innovation lies in the focal loss, which allows the network to focus more on the misclassified samples.

Models such as YOLO [22] and SSD [13] that are without a sampling process suffer from an overwhelming number of candidate locations, most of which clearly do not contain an object. This causes an imbalance, as "easy" examples dominate the gradient and prevent the network from properly learning from the more difficult examples. As a result, training is less efficient and the model is weaker. Focal loss reduces impact of the examples where the model is confidently correct, so that the impact of the examples where the model struggles is proportionally larger.

Mathematically, focal loss is an extension of the cross entropy loss $-\log(p_t)$, where p_t is defined as the probability assigned to the correct class by the model. A "modulating factor" $(1-p_t)^\lambda$ is used, so that at λ greater than 1, when the probability assigned to the correct class is low, the factor evaluates to one, while when the probability assigned to the correct class is high, the factor evaluates to zero, signifying that there is no longer need to focus on this example as much. In summary, the focal loss can be expressed as $L(p_t) = -(1-p_t)^\lambda \log(p_t)$.

The input to RetinaNet is flexible and dependent on the type of feature extractor used but it prefers images of size 800×1333 or less. RetinaNet uses a FPN (Feature Pyramid Network) [9] on a ResNet-101 [7] as its feature extractor, a standard feature extractor for object detection. Then, it has two branches to predict the bounding boxes (of output size $W \times H \times KA$) and the classes respectively (of output size $W \times H \times 4A$) where W and H are the width and height at that stage of the feature pyramid, A is the number of anchors, and K is the number of classes.

4.3. YOLOv3

YOLOv3 [24] is an enhancement on the YOLO [22] ("you only look once") series of models. The YOLO framework is a deliberately end-to-end one-stage model that has advantages in simplicity, speed, and the amount of contextual information. It removes the region proposal step found in 2-stage object detectors by using the features of the entire image and using a grid system where each cell of the grid gets a certain number of box predictions (where each prediction includes the x-coordinate, y-coordinate, width, height, and confidence, as well as class probabilities). YOLOv3 is an update to previous YOLO models

(YOLO and YOLOv2 [23]) that includes a more powerful feature extractor with residual connections and feature pyramids. YOLOv3 works optimally at an input resolution of 416×416 pixels due to its grid structure, but can take a wide range of inputs of any resolution (however, it then resizes those inputs so the longest side is 416 pixels). Its output is of shape $19 \times 19 \times 3 \times (5 + K)$, where 19×19 comes from the grid format of YOLO while 3 comes from the 3 layers of the feature pyramid and $5 + K$ comes from the 4 bounding box coordinates plus an objectness probability and K class probabilities.

4.4. Neural Style Transfer (MSG-Net)

Neural style transfer is a technique that is used to combine the content of one image with the style of another image. Multi-style Generative Network (MSG-Net) [32] is an improvement on the original neural style transfer algorithm that introduces a 4-dimensional (instead of 2-dimensional) Gram matrix called the "CoMatch" layer to increase expressivity of the network and increase detail of the generated image.

4.5. Transfer Learning

We leveraged transfer learning to help the model learn a better reepresentation of the image data. In particular, we used Faster R-CNN, RetinaNet, and YOLOv3 modeels that were pretrained on the COCO database [11] and then fine-tuned their parameters on our downstream task.

4.6. Baseline Model

For the baseline comparison, we used the YOLOv2 model [23], which was tested in Ogawa et al [19]. The YOLOv2 model has the same input format as the YOLOv3 model (explained in the above section) and outputs a $13 \times 13 \times (5 + K)$ shape tensor where 13×13 is the grid size used by YOLOv2 and $5 + K$, like in YOLOv3, represents the 4 bounding box coordinates, an objectness probability, and K class probabilities.

5. Implementation

We developed our code in PyTorch [20] and Torchvision [15]. Our models were built on top of existing implementations of Faster-RCNN [21], RetinaNet [15], YOLOv3 [12], and neural style transfer with MSG-Net [31]. We also used the Manga109 API [14] in our data preprocessing pipeline. The code for our project can be found at https://github.com/taivu1998/MangaObjectDetection.

Model	mAP
YOLOv2 (baseline) [19]	59.7
Faster R-CNN	57.4
Faster R-CNN (TL)	62.0
Faster R-CNN (TL + DA)	64.5
Faster R-CNN (TL + NST1)	55.4
Faster R-CNN (TL + NST2)	56.4
RetinaNet	57.4
RetinaNet (TL)	60.8
RetinaNet (TL + DA)	63.5
YOLOv3 (TL + DA)	**71.0**

Table 1. Results of our models compared to the baseline

6. Experiments

6.1. Experiment Setups

The Faster R-CNN model and the RetinaNet model were trained for 10 epochs. We experimenting with training them from scratch and fine-tuning models that were pretrained on ImageNet. We optimized it using Adam optimization algorithm with a fixed learning rate of 0.0001 (as this learning rate gave the best performance during our hyperparameter search) and a batch size of 8 (due to limited GPU memory).

We used a YOLOv3 model that was pretrained on ImageNet and fine-tuned for 10 epochs. We used the Adam optimizer with a learning rate of 0.001 and we used a batch size of 16.

6.2. Evaluation Metric

Since this project addressed an object localization task, we utilized Mean Average Precision (mAP) [4] as our quantitative evaluation metric.

To define mAP, we use the concept of Intersection over Union (IoU), which is the ratio between the intersection and the union of a predicted box and the corresponding ground truth box. If IoU is higher than a given confidence threshold (such as 0.5), we get a true positive, otherwise the box is considered a false positive. All the objects that the model misses are consider false negatives. By using this information, we can compute the precision and recall scores of the model's predictions.

Subsequently, we select 11 different confidence thresholds corresponding to 11 equally spaced recall levels $0, 0.1, 0.2, ..., 1$. The mAP score is defined as the average of the precision values at these 11 thresholds.

6.3. Results and Discussion

6.3.1 Differences in Performance through Model Architectures

We observe that without transfer learning and data augmentation, FasterR-CNN and RetinaNet achieved similar levels

of performances, with both scoring a mAP of 57.4. With transfer learning, FasterR-CNN (with a mAP of 62.0) scores slightly better than RetinaNet (with a mAP of 60.8), and with transfer learning and data augmentation, FasterR-CNN (with a mAP of 64.5) is also slightly better than RetinaNet (with a mAP of 63.5). In the big picture, the differences between the mAP scores for FasterR-CNN and RetinaNet are not that large, so they could be said to have similar performance on this task of manga object detection.

We think that the similar performance comes from RetinaNet's lack of focus on the easy examples. We originally theorized that RetinaNet (due to its focal loss) would perform significantly better than FasterR-CNN, as our dataset contains many easy examples (e.g. the frames of the manga) and some harder ones (text, because they are small). However, we think that because of this very reason, RetinaNet does not perform better than FasterR-CNN. Since a sizable proportion of the objects in the dataset are frames (the easy examples), a lack of focus on these examples due to the focal loss may cause RetinaNet to get some of these wrong, offsetting the benefits that it gets from focusing on the harder examples. Meanwhile, while FasterR-CNN may get some of the harder examples wrong, it might be more accurate on the easy examples, making the mAP look similar for both architectures.

We also observe that with transfer learning and data augmentation, YOLOv3 (with a mAP of 71.0) performs far better than FasterR-CNN (with a mAP of 64.5). We think that the YOLOv3 achieved better outcomes than Faster-RCNN due to the precence of overlapping objects in many examples. In fact, these overlapping objects confuses the regional proposal process in Faster-RCNN, making less efficient in separating and recognizing each object individually. Originally, we theorized that FasterR-CNN would perform better because YOLOv3 was a one-stage detector and thus leaned more towards speed than accuracy compared to Faster-RCNN, a two-stage detector. However, we underestimated the sheer efficiency of the DarkNet53 backbone [24].

We also see that with transfer learning and data augmentation, YOLOv3 (with a mAP of 71.0) performs far better than RetinaNet (with a mAP of 63.5). This result is interesting because YOLOv3 has similar performance to RetinaNet in [24]. It may be because YOLOv3, unlike the previous version of YOLO, no longer struggles with small objects (in our case, some 'text' and 'face' objects). Also, mangas are very context-dependent, so YOLOv3 with a wider context can be able to consider more information and thus become more expressive.

6.3.2 Transfer Learning

We observe that the addition of transfer learning has a beneficial effect on model performance. Without transfer learning, our Faster R-CNN model achieved a mAP score of 57.4 on the validation set, which was quite close to but under the baseline outcome of 59.7. With transfer learning, the Faster R-CNN model achieved a mAP score of 62.0 (4.6 higher than without transfer learning). Similarly, without transfer learning, the RetinaNet model had a mAP score of 57.4, but with transfer learning, the RetinaNet model had a mAP score of 60.8 (3.4 higher).

This significant increase in performance is likely because the pretrained models have learned better and perhaps more general representations of images. We suspect this to be the case as manga images are more similar than different, and as such, are less able to teach models about general image content and structure. Furthermore, the model that is trained from scratch has to learn both the basic image structure and the nature of the objects to detect, while in the same number of epochs the pretrained model only has to learn about the nature of the objects to detect, making it more difficult for the model trained from scratch to get up to speed on the task.

6.3.3 Data Augmentation

We can see that the addition of data augmentation has a beneficial effect on model performance. Our Faster R-CNN with transfer learning achieves a mAP score of 62.0, but our Faster R-CNN with transfer learning and data augmentation achieves a mAP score of 64.5 (2.5 higher). Moreover, our RetinaNet with transfer learning gets a mAP score of 60.8, while our RetinaNet with transfer learning and data augmentation achieves a mAP score of 63.5 (2.7 higher).

This significant increase in performance may be due to the increased robustness of the models with data augmentation. Manga images all have a similar general structure, but there may be some images that are outliers to this structure. For one, different mangas are drawn with different average brightness levels, so doing augmentations on brightness during training may induce the model to perform at its best level during all kinds of brightnesses when testing.

6.3.4 Style Transfer

We observe that style transfer had a detrimental effect on model performance. Our FasterR-CNN with transfer learning but without style transfer had a mAP of 62.0, while our FasterR-CNN with transfer learning and style transfer with style 1 had a mAP of 55.4 (6.6 lower) and our FasterR-CNN with transfer learning and style transfer with style 2 had a mAP of 56.4 (5.6 lower).

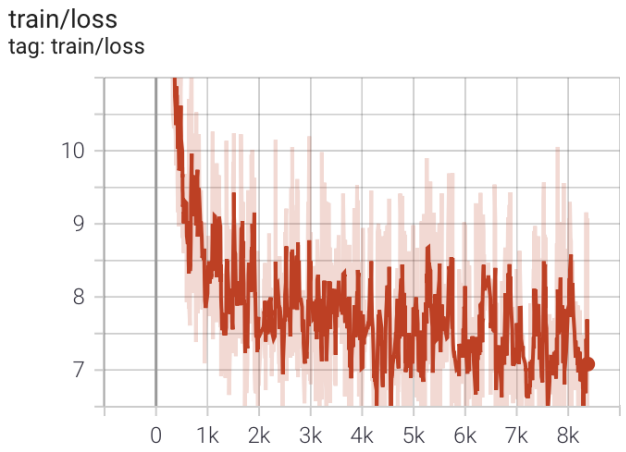

Figure 3. YOLOv3 train loss. X axis denotes the iteration number and Y axis denotes the total training loss.

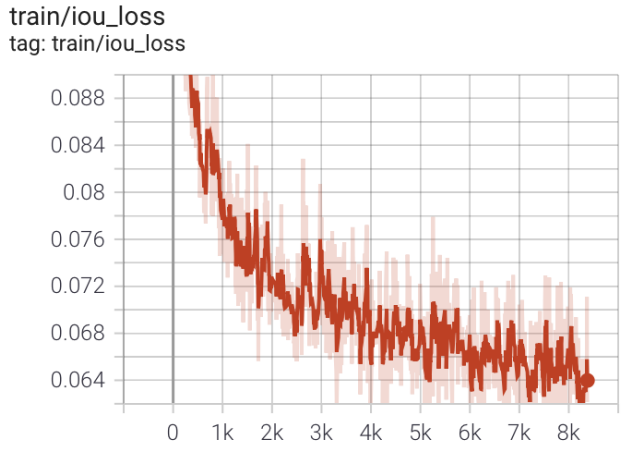

Figure 4. YOLOv3 IOU loss. X axis denotes the iteration number and Y axis denotes the IOU loss.

We think that the significant decrease in performance comes from the addition of insignificant information and the loss of important information due to the style transfer. We originally theorized that style transfer would solve two of the three major difficulties in object detection on mangas, namely the large amount of whitespace in manga sketches and the predominant grayscale tone of most mangas, since it ameliorates the whitespace concerns by adding texture to all spaces and partially fixes the grayscale tone through colorization. But, perhaps, the textural information that the style transfer adds to the image may actually be irrelevant to the image since it comes from another image, meaning that the new textures, although covering the whitespace, provides no new information. Meanwhile, the image may lose information as style transfer mixes pre-existing textures with its own. For example, it may be harder to detect text when the unique texture of text is overwritten by wavy lines.

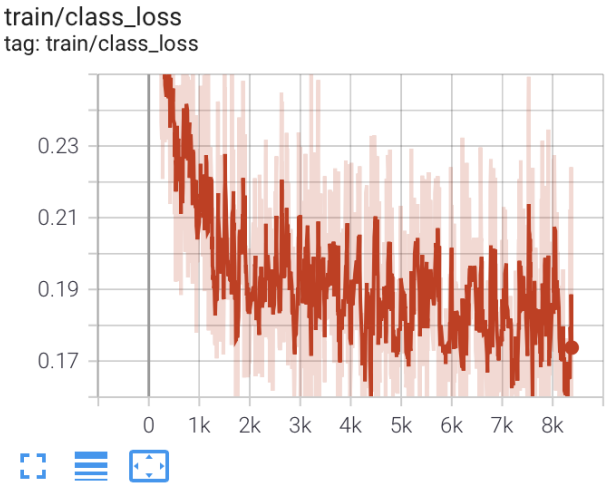

Figure 5. YOLOv3 train class loss. X axis denotes the iteration number and Y axis denotes the class loss.

6.3.5 YOLOv3: Analysis of Training

In this section we analyze the training and validation of YOLOv3, our best-performing model. For this analysis, we trained YOLO an extra 5 epochs in order to better see the trajectory of the plots for both the train loss and val mAP 7. The loss has three components: IOU, class, and objectness. As seen in figure 6, we see that the objectness classifier learns the fastest, as the objectness loss drops to near the minimum within 1000 iterations (approximately 2 epochs). Whereas, as seen in figure 4, the IOU predictor trains a lot slower. This is because predicting the IOU is a lot more difficult than predicting the objectness score. The class loss plots in figure 5 and the overall loss in figure 3 show similar shapes and gradually decrease, which is a good sign.

We see that the val mAP increases even past 10 epochs. However, the rate of increase seems to be slow down, and the validation mAP curve seem to be plateau, while the training loss keeps decreasing. We suspect that our model starts to overfit to the training data at this point.

6.3.6 Qualitative Analysis

Looking at the outputs of the models gives us more insights into how they distinguish and localize different objects. In general, our models achieve good performance in finding objects in comic pages, drawing bounding boxes around them, and classifying their types. As shown in Figure 8, when the objects are well separated, our models were able to localize them correctly. In addition, the models performed extremely well in recognizing the presence of frames and

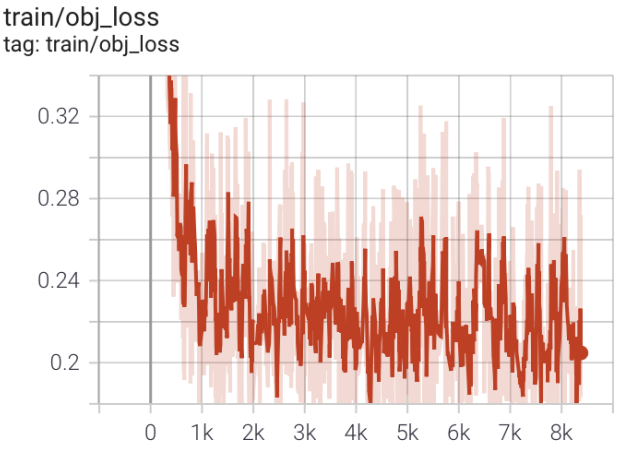

Figure 6. YOLOv3 train objectness loss. X axis denotes the iteration number and Y axis denotes the objectness loss.

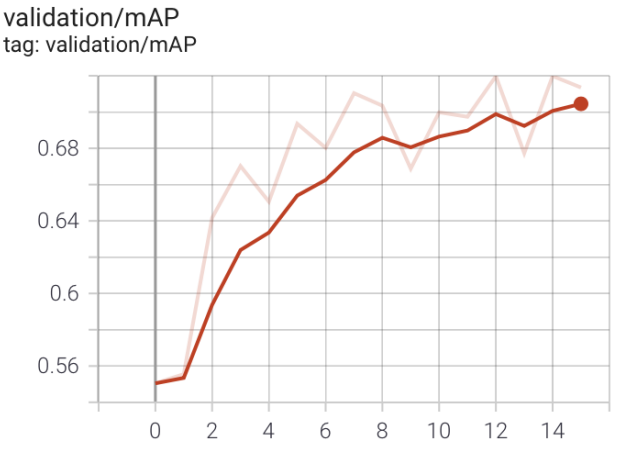

Figure 7. YOLOv3 val mAP scores. X axis denotes the epoch number and Y axis denotes the mAP score.

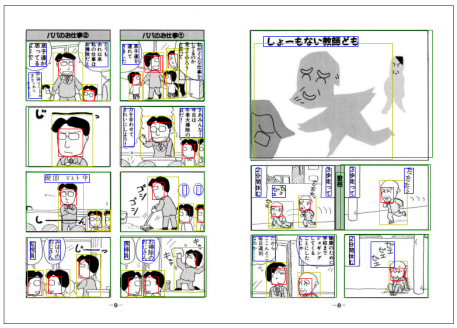

Figure 8. An example where the model perform well

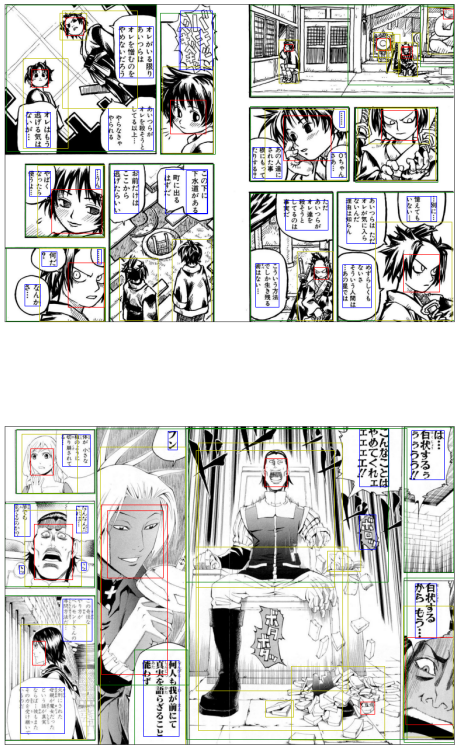

Figure 9. An example where the model fails to detect text

texts (see Figures 8, 9, and 10). This might bee because the frames are normally rectangular with clear structures, while characters' faces have noticeable patterns and are separated from each other.

However, the outcomes for the "text" class seem a bit less desirable (see 9). While the models succeeded in both cases, they sometimes failed to recognize texts because the speech balloons have a color other than white (such as gray color). When several speech balloons were too close to each other, the models were not able to separate them and instead consider them as one single text balloon. Similarly, as shown in Figure 10, the models make errors in detecting bodies when there are a lot of close human bodies in a single panel. There was also an interesting case where the model got confused by non=human objects like vehicles, and assign the "body" label to them.

7. Conclusion & Future Work

In summary, we developed and compared several models on the task of object detection on mangas. The best model was YOLOv3 with transfer learning and data augmentation, which attained 70.1 mAP on the Manga109 dataset. Many of our models (FasterR-CNN TL, Faster R-CNN TL+DA,

Figure 10. An example where the model fails to detect body

RetinaNet TL, RetinaNet TL+DA, and YOLOv3 TL+DA) beat the baseline of 59.7 mAP. We also tried neural style transfer as a featurizer targeted specifically toward sketches through its properties, but found through experimentation that it was not an effective featurizer.

We acknowledge that the choice of style images may affect the performance of the neural style transfer featurized FasterRCNN. In the future, we can continue to try a more diverse array of style images. We can also try fusing other types of features (such as the Histogram of oriented gradients (HOG) [17], which has also been suggested as a possible featurizer for object detection) either with the output from the backbone or the output before the final layer of the second stage. In regards to FasterR-CNN [25], RetinaNet [10], and YOLOv3[24], we will further tune the hyperparameters and perform architecture search to increase performance, and we can also try other model architectures such as EfficientDet [29].

8. Contributions & Acknowledgements

Robert worked on developing the Neural Style Transfer featurizer methodology (including preprocessing, developing the pipeline through the NST algorithm, and training Faster-RCNN with NST). He also worked on preprocessing for YOLO and training YOLO. Tai worked on preprocessing the Manga109 dataset, implementing the data pipeline, building the FasterRCNN and RetinaNet models, training and tuning the models, and implementing various forms of data augmentation. Robert and Tai both worked on writing the report and planning the overall structure of the project.

As stated in Section 5, we made use of the following frameworks and public code resources:

- PyTorch: https://pytorch.org/
- Torchvision: https://github.com/pytorch/vision
- TorchVision Object Detection Fine-tuning Tutorial: https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html
- PyTorch-YOLOv3: https://github.com/eriklindernoren/PyTorch-YOLOv3
- PyTorch-Style-Transfer: https://github.com/zhanghang1989/PyTorch-Multi-Style-Transfer
- Manga109 API: https://github.com/manga109/manga109api

References

[1] Kohei Arai and Herman Tolle. Method for real time text extraction of digital manga comic. *International Journal of Image Processing (IJIP)*, 4(6):669–676, 2011. 2

[2] Wei-Ta Chu and Ying-Chieh Chao. Line-based drawing style description for manga classification. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 781–784, 2014. 2

[3] Wei-Ta Chu and Wei-Chung Cheng. Manga-specific features and latent style model for manga style analysis. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1332–1336. IEEE, 2016. 2

[4] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 4

[5] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1, 3

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[8] Jochen Laubrock and Alexander Dunst. Computational approaches to comics analysis. *Topics in cognitive science*, 12(1):274–310, 2020. 1

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017. 3

[10] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1, 2, 3, 8

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 4

[12] Erik Linder-Norén. Pytorch-yolov3, 2021. 4

[13] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016. 3

[14] manga109. Manga109 api, 2018. 4

[15] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010. 4

[16] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 76(20):21811–21838, 2017. 1, 2

[17] Robert K McConnell. Method of and apparatus for pattern recognition, Jan. 28 1986. US Patent 4,567,610. 8

[18] Nhu-Van Nguyen, Christophe Rigaud, and Jean-Christophe Burie. Comic characters detection using deep learning. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 3, pages 41–46. IEEE, 2017. 2

[19] Toru Ogawa, Atsushi Otsubo, Rei Narita, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. Object detection for comics using manga109 annotations. *arXiv preprint arXiv:1803.08670*, 2018. 1, 2, 4

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. 4

[21] PyTorch. Torchvision object detection fine-tuning tutorial, 2021. 4

[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 3

[23] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016. 4

[24] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018. 2, 3, 5, 8

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016. 1, 2, 3, 8

[26] Christophe Rigaud, Norbert Tsopze, Jean-Christophe Burie, and Jean-Marc Ogier. Robust frame and text extraction from comic books. In *International Workshop on Graphics Recognition*, pages 129–138. Springer, 2011. 2

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[28] Weihan Sun, Jean-Christophe Burie, Jean-Marc Ogier, and Koichi Kise. Specific comic character detection using local feature matching. In *2013 12th International Conference on Document Analysis and Recognition*, pages 275–279. IEEE, 2013. 2

[29] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 8

[30] Christopher Thomas and Adriana Kovashka. Artistic object recognition by unsupervised style adaptation. In *Asian Conference on Computer Vision*, pages 460–476. Springer, 2018. 2

[31] Hang Zhang. Pytorch-style-transfer, 2017. 4

[32] Hang Zhang and Kristin Dana. Multi-style generative network for real-time transfer. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 4