AUGMENTING BIOLOGICAL PATHWAY EXTRACTION WITH SYNTHETIC DATA AND ACTIVE LEARNING

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

JOSHUA LEE THOMPSON

Thesis Supervisor: Dr. Dong Xu

May 2022

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled:

AUGMENTING BIOLOGICAL PATHWAY EXTRACTION WITH SYNTHETIC DATA AND ACTIVE LEARNING

presented by Joshua Lee Thompson, a candidate for the degree of Master of Science and hereby certify that, in their opinion, it is worthy of acceptance.

____

Dr. Dong Xu

____

Dr. Mihail Popescu

____

Dr. Ye Duan

ACKNOWLEDGEMENTS

I would like to sincerely thank Dr. Dong Xu for his support over the last 4 years. Dr. Xu provided me the space and resources to grow more academically, professionally, and personally than I ever thought possible. I also want to thank Dr. Fei He for being an amazing mentor and role model to me over the past 4 years. His guidance and help have been instrumental for the success of this thesis. My work would also not have been as well developed without the valuable insights from Dr. Mihail Popescu. I also want to thank Dr. Ye Duan for his interest in my studies and being a part of my thesis committee.

Most importantly, I want to thank all of my friends and family for their unwavering support over the past challenging two years. I especially want to thank Karlye Thompson, Edgar Arroyo, Gavin Phillips, and Joshua Love. They were always there for me when I needed them the most and provided a safe, secure foundation that I was able to build from. Without these four, I would not have been able to complete this thesis. I truly love and appreciate them all.

Joshua Thompson

Table of Contents

List of Tables

List of Figures

ABSTRACT

The corpus of biomedical literature is growing rapidly as many papers are recorded in PubMed every day. These papers often contain high-quality biological pathways in their figures/text, which are great resources for studying biological mechanisms and precision medicine. However, it can take a long time for many of these works to be put into practical use as each paper's contributions need to be curated by experts. This, often lengthy, process causes professional practice to lag behind research. To speed up this process, I helped develop a pipeline that integrates NLP and object detection processing to extract gene relationships reported in articles' figures and text. This pipeline was able to extract such relationships with high precision and recall on a small, annotated set. However, extending this pipeline for improved generalization and new settings was limited by the number of high-quality annotations available. Such labeled data is very time consuming to collect and traditional augmentations were observed to generate diminishing returns. To address this shortcoming, I developed an approach for generating purely synthetic data for object detection on biological pathway diagrams based on a set of rules and domain knowledge. Our method iteratively generates each pathway relationship uniquely and is demonstrated to improve the generalization of our object detection model significantly across a variety of settings.

Additionally, with the capability to generate unique and informative samples, we integrated our synthetic generation methodology into an active learning setting. While traditional active learning relies on a pool of unlabeled data to draw from with an acquisition function, our method is pool-less and does not require any acquisition function. Instead, we generate each batch of data uniquely based on the training losses from the previous batch. Pool-less Active Learning (PAL) via synthetic data generation is demonstrated to reduce the number of iterations required for model convergence during training on pathway figures.

Chapter 1

Introduction

In this chapter I will discuss the motivations for my work, the significance of my research, the specific issues I addressed, and how I went about solving those problems. I will also outline the organization for the rest of this thesis.

1.1 Motivations

The corpus of biomedical literature is growing rapidly as many papers are recorded in PubMed every day. These papers often contain high-quality biological pathways in their figures/text, which are great resources for studying biological functions and precision medicine (Figure 1.1). However, it can take a long time for many of these works to be put into practical use as each paper's contributions need to be curated by experts. This, often lengthy, process causes professional practice to lag behind research. To speed up this process, I helped develop a pipeline that integrates natural-language-processing (NLP) and object detection to extract gene relationships reported in articles' figures and text. This pipeline was able to extract such relationships

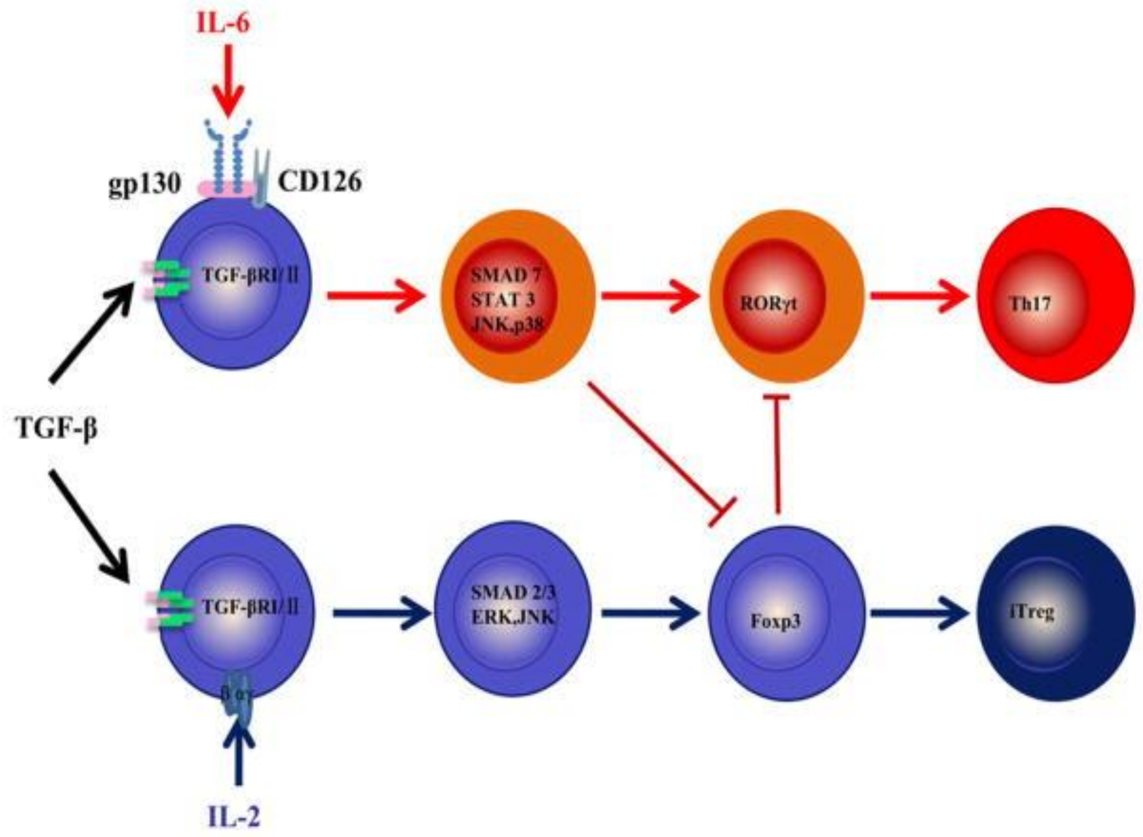


Figure 1.1: Example of a signaling pathway pulled from [1] detailing the role of TGF-β in iTreg and Th17 cell development. Text entities in these figures are genes or other biological components. Arrows represent gene activations and t-bars indicate gene inhibitions.

with high precision and recall on a small, annotated set. In addition to speeding up curation efforts, such extraction has the potential to enable novel knowledge discovery by mapping previously overlooked gene- and drug-pathway interactions across the literature. Being able to extract such new and long-range connections has wide applicability to precision medicine practice as well.

However, extending this pipeline to new settings was limited by the number of high-quality annotations available. This is a frequently occurring problem in the biomedical field and other data hungry areas where high quality labeling is time consuming to collect and traditional augmentations generate diminishing returns. To address this shortcoming, I helped develop an approach for generating purely synthetic data for object detection on biological pathway diagrams based on a set of rules and domain knowledge. This method iteratively generates each pathway relationship uniquely and is demonstrated to improve the generalization of our object detection model significantly across a variety of settings. Such a generative method could have applications in other document, figure, and diagram analysis tasks as well (such as OCR [2] or Document-Analysis [3]) where labeling often bottlenecks development.

With the capability to generate unique and informative samples, we further integrated our synthetic generation methodology into an active learning setting. We did this to get the most value out of each training iteration. While traditional active learning relies on a pool of unlabeled data to draw from with an acquisition function, our method utilizes no data pool and does not require any acquisition function. Instead, we generate each batch of data uniquely based on the training losses from the previous batch. This new approach for active learning marks a step towards further integrating data augmentation with active learning.

1.2 Problem Statement

The goal of my thesis is to answer the following questions relating to biological pathway extraction:

1. How to extract relationships from pathway figures?
2. How to integrate different output sources to filter gene relationships?
3. How to up-sample our figure dataset with unique training signals?
4. What are the limits to our up-sampling approach?
5. How to effectively leverage rule-based up-sampling during training?

1.3 Contributions

The main contributions of this thesis can be summarized as follows:

1. A unified pipeline for extracting gene interaction triplets from biomedical articles that combines image and natural language processing.

2. A novel rule-based algorithm for generating annotated pathway diagrams.
3. Characterizing good practice for training with synthetic and real data.
4. Measuring the impact of structured and unstructured noise for synthetic pathways.
5. Pool-less Active Learning (PAL) via synthetic data generation.

1.4 Thesis Organization

This thesis is organized into six Chapters. Chapter 1 introduces my research problem and the motivations for my work. Chapter 2 outlines a detailed literature review of works that I took inspiration from. Chapter 3 highlights the development of our unified pipeline for gene relationship extraction. In Chapter 4, I introduce my novel rule-based method for up-sampling annotated diagrams with a purely synthetic approach. Chapter 5 investigates how to best leverage our ability to generate fully synthetic samples that generalize, by integrating our augmentation method into an active learning framework. Chapter 6 reflects on my work and suggests future directions for further investigation.

Chapter 2

Literature Review

In this chapter, I will review several fields related to my work and other methods that shaped my development. Specifically, this literature review will discuss related works in the areas of gene-gene relationship extraction, object detection, data augmentation, and active learning.

2.1 Gene Relationship Extraction

As previously mentioned, biomedical literature publishes biological pathways at a rapid pace. These pathways, presented in text and image formats, are great resources for studying biological functions and precision medicine practice. For example, the most up-to-date knowledge about newly discovered non-canonical disease pathways and uncommon drug actions is vital in studying patient-specific biomolecular phenotypes for cancer treatment [4]. However, to effectively use large scale pathway information, new pathways from literature need to be carefully curated, reconciled, and transformed into a computable form [5]. Manual curation and text mining are the two main approaches employed for this task. Kuenzi et al. manually curated 2,070 cancer pathways to identify previously underappreciated functions and discover new genes to known cancer pathways [6]. However, PubMed’s library continues to grow by more than a million articles per year [7], which is unmanageable by manual curation alone. While there are advances in text mining approaches for extracting simple biological interactions, such as protein-protein interactions and biological events that include two or three such interactions, there are no reliable methods to extract larger, more complex disease pathways, such as signaling or drug action pathways [8]. One challenge remaining is that references between biological entities can be spread far across the text and cannot be easily reconciled [9, 10]. For instance, information extraction tools for biological events had error rates from 23% to 58% [11]. Additionally, the error rates for chemical-induced disease relation extraction methods ranged between 43% to 68% [12]. Coreferences and anaphoric expressions are also still challenging for text mining tools alone [8].

To address this challenge, my team observed that nearly all articles on newly discovered pathways contain figures that summarize their findings. To best leverage these figures, we proposed an integrated bio-curation pipeline for mining genes and their interactions from pathways by jointly utilizing an article’s figures and text. We hypothesized that the extraction of genes and their interactions from pathway figures and text will be more accurate and reliable than extraction from either alone.

Information extraction from images is a new direction in biological curation. Recently, only a few studies have been conducted to extract genes from publication figures using optical-character-recognition (OCR). Different extensions and improvements for such OCR have been applied for image segmentation, localization, and recognition tasks from biological image text in [13-15]. Additionally, in a large-scale analysis of pathway figures [16], gene names from images of were retrieved but the interactions between them were ignored. Even though extraction of biological relationships from images was previously described [17], there were no details for reproducibility or accessible online resources. Our early study [18] demonstrated that it was feasible to retrieve gene names and gene relationships from pathway figures.

2.2 Object Detection

Looking towards object detection, much of the progress in this field from the past decade can be attributed to improved architecture design. RCNN [19] was the first network to apply high-capacity convolutional neural networks to bottom-up region proposals in order to localize and segment objects from images. They did this by generating region proposals from an image, extracting a fixed length feature vector from each region, passing that vector through a convolutional neural network feature extractor, and using a set of class specific linear support vector machines to classify each proposal. Fast-RCNN [20] took this process one step further and used the entire image as input to the convolutional neural network. They did this to share the single feature map across proposals to save on computational overhead. To predict for any given proposal, they then used region of interest pooling to extract a fixed-length feature vector. This feature vector was then fed through several fully connected layers that branch off to localize an object and predict its class. Faster R-CNN [21] further unified this design by using the features generated from the input image to also calculate the proposals with a region proposal network. They did this because generating good object proposals was often a computational bottleneck and a trade-off between accuracy and good performance. Feature Pyramid Networks [22] were then designed to address poor recognition of objects at different scales. They use feature pyramids built from image pyramids to capture different sized objects by leveraging top-down and lateral skip connections. Predictions are then made at each level of the pyramid. RetinaNet [23] introduced another single-stage detector to leverage robust feature extraction with ResNets [24], Feature Pyramid Networks, and fully convolutional networks for regression and classification. However, the main contribution of this paper [23] was their introduction of the focal loss, a modified cross entropy function designed to address imbalance between classes and hard training samples. Additionally, many more object detection architectures have been proposed with a wide range complexity: YOLO [25], SpineNet [26], DETR [27], etc.

2.3 Data Augmentation

Training such large object detection models typically requires large amounts of data as well. However, in many areas high quality ground truth labels can be expensive to collect. As such, many practitioners and researchers alike often turn towards data augmentation methods to increase their training pool size. Traditional data augmentations, especially for images, usually focus on positional modifications such as random flips, scaling, cropping, rotations, translations, etc. [28]. This kind of augmentation is helpful for making models capture similar signals, but from different viewpoints. Other types of augmentation will focus on changing color characteristics such as lighting, contrast, hue, and saturation [29]. This class of augmentation is helpful to make models more color agnostic and focus on shape features. Involved modifications have also been shown to be helpful such as kernel filters, random erasing (CutOut [30]), injecting random noise, and mixing samples (MixUp [31] and CutMix [32]). These methods can promote a model to use more contextual information and leverage larger-scale features. An overview of these methods is provided in [33]. The unifying goal of these methods is to create new training samples by manipulating pre-collected data. However, it has been observed that this can lead to diminishing returns as the same or similar signals are repeatedly seen during training.

Another approach to up-sampling does not just modify existing images but instead creates entirely new ones. This approach often leverages deep learning methods such as conditional GANs [34,35], Variational Autoencoders [36], Spatial Transform Networks [37] or neural style transfer [38]. Synthetic data generation has also been previously applied for generating 3D point clouds for training in [39,40]. Our work is complementary to other synthetic data generation methods and targets the object detection task specifically based on a set of rules. Our method is less expensive and more biology-aware than related deep learning approaches.

2.4 Active Learning

With an abundance of unlabeled samples in big data analysis, active learning has become a growing area of research in recent years. Active learning not only focuses on learning from data, but also learning what data to learn from. Active learning primarily focuses on prioritizing learning from data that will have the most impact during training. This is done by first labeling a very small subset of a large dataset manually and then training on that subset. The goal of this initial training is to better understand which areas of the parameter space need to be labeled. After training, the model is used to predict each un-labelled sample and a priority score is calculated for it. Based on this priority score, a new subset is selected for labeling and is added to the growing training set for the next round of training. This process is repeated several times to continuously update the training set with the goal of improving generalization without having to annotate the entire dataset. This technique can be very useful when you have large amounts of unlabeled data to train with. However, selecting the best priority score metric is often problem specific. A least confidence priority score takes the highest probability for each sample's prediction and sorts samples from smallest to largest [41,42]. Samples with the lowest-confidence predictions are selected to be trained on. Margin sampling takes into account the difference between the highest predicted probability and the second highest probability [43]. The intuition is that we want to effectively discriminate between the two most likely classes. Then, to do that, we prioritize points that have the hardest time choosing between their top two classes. Another option is to use entropy, which is similar to margin sampling, but more holistic [44]. Using entropy promotes discrimination between all classes by prioritizing points that have trouble ruling out many classes and those that are not very confident at all. Other approaches include Query by committee [45], Monte Carlo Dropout [46], BALD [47], and Learning to predict the loss [48].

One of the challenges in selecting an acquisition function involves balancing uncertainty and diversity sampling. Uncertainty sampling aims to use estimates of what the model is uncertain about as analogs for what the model would be wrong about. Diversity sampling tries to get a balanced training set from samples that are not yet annotated. Some other methods try to balance the two [49-51]. Overviews of active learning and applications thereof are provided in [52,53].

Another area of interest combines data augmentation with active learning. Intuitively, both are trying to get the most use from each sample during training. However, how to best combine the two methods is an open research topic. GAN Data Augmentation Through

Active Learning Inspired Sample Acquisition [54] used a MNIST trained GAN to up-sample the MNIST dataset as their data pool for active learning. Deep Active Learning with Augmentation-based Consistency Estimation [55] illustrated how CutOut and CutMix augmentation can be used as uncertainty measures, how these uncertainty measures can be used as priority metrics, and how these measures can be used as general regularization terms. Bayesian generative active deep learning [56] showed the benefit of augmenting samples selected from the acquisition function. Look-Ahead Data Acquisition via Augmentation for Deep Active Learning [57] took this approach one step further by jointly considering unlabeled samples and their augmentations in the acquisition stage. My contributions in active learning are complementary to this line of research that combines data augmentation with active learning. Specifically, I introduce Pool-less Active Learning (PAL) via synthetic data generation that does not need any previously collected data or an acquisition function.

2.5 Literature Review Summary

In this chapter, I presented an overview of previous attempts to extract biological relationships from text and figures. I also described the improvements made to recent object detection methods that I leverage in later chapters. I also outline several drawbacks of many traditional data augmentation methods and a new line of research that tries to overcome these disadvantages. Additionally, I introduced the active learning training procedure and how new methods try to incorporate data augmentation with active learning. In the next chapter, I will detail the pipeline I developed to extract gene-gene relationships.

Chapter 3

Gene-Interaction Extraction

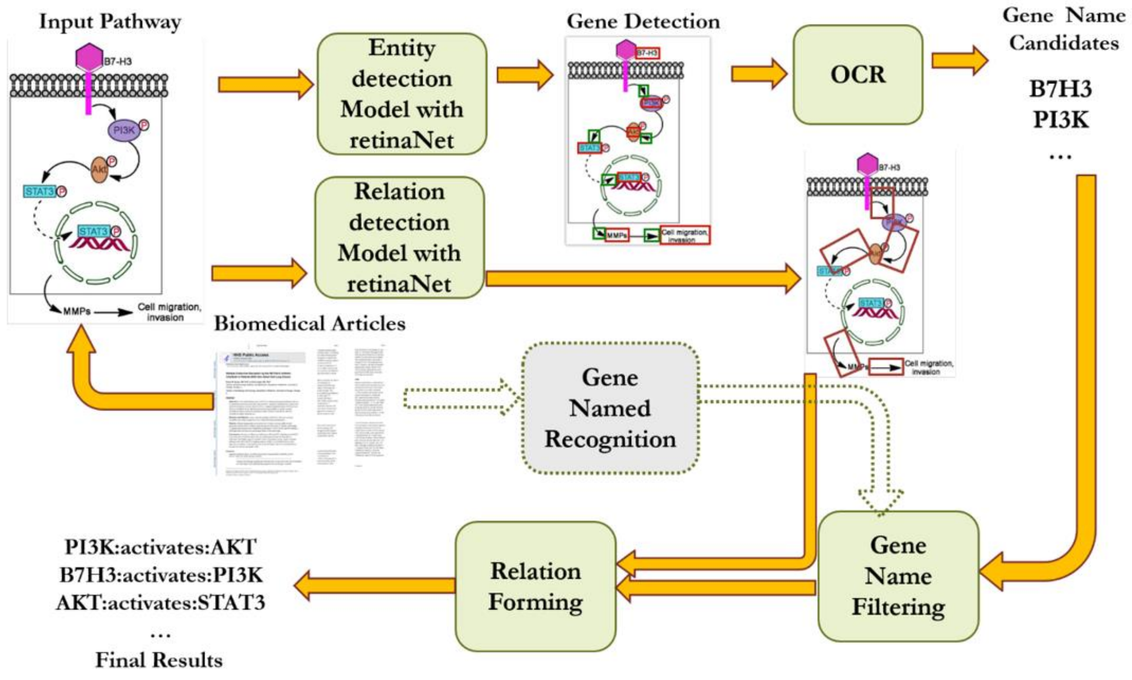


Figure 3.1: Overview of our gene interaction extraction pipeline.

In this chapter I outline the motivations for our relationship extraction pipeline design, detail our implementation, and report our experimental results.

3.1 Mixing Modalities

To visualize biological functions, article diagrams often use simple shapes and indicators to make the relationships between entities clear. To extract gene names from these figures, one may intuitively think to just apply optical character recognition (OCR) to extract entity text. However, in practice these results may not always be correct due to various artifacts in the images. Fortunately, article text offers detailed information to correctly identify entity names. This information can then be used to filter out or correct mislabeled entity names from figures. While very useful for extracting gene names, it remains difficult to precisely extract the relationships between objects from the text alone. We hypothesize an integrated approach between text and image processing can overcome the limitations each extraction method may face alone.

In this study, we designed a deep learning-based pipeline to detect genes and their interactions from pathway figures and utilize text information from articles to filter the results. As a preliminary attempt in pathway curation, we only focused on extracting genes and two key types of gene interactions (activation and inhibition). These indicators are usually plotted near textboxes with simple arrows and T-bar lines. Other types of interactions are explored in the next chapter.

To the best of our knowledge, there was no previous work done to systematically extract biological mechanisms from figures.

Figure 3.1 illustrates an overview of our pipeline, which includes four main components: (1) two object detection models to locate the genes and the indicators defining gene interactions from pathway figures; (2) an OCR module to convert text regions into computable gene names, (3) a gene name filtering module to only keep valid gene names from all results, and (4) a gene-interaction prediction module to connect pairs of genes in a recognizable interaction.

3.2 Object Detection

For this study, we used the RetinaNet [23] architecture to detect all text regions and interaction indicators from the pathway figures pulled from articles. RetinaNet was chosen since it can achieve a good computational complexity-accuracy trade-off. This model is composed of three modules: a backbone network, a FPN (Feature Pyramid Network), and detection heads. The architecture of RetinaNet is shown in Figure 3.2.

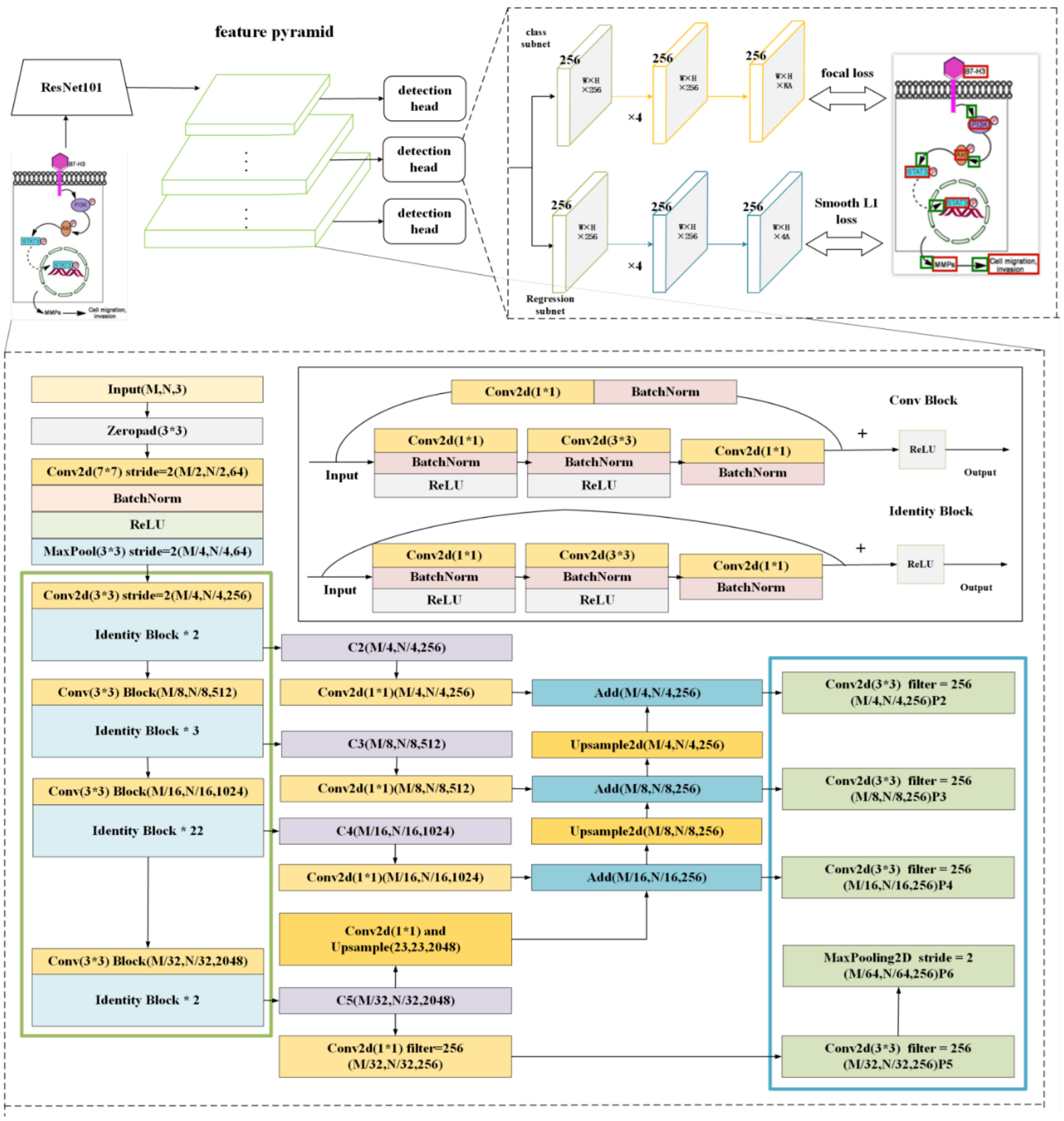


Figure 3.2: Over of the RetinaNet-101 architecture. This model has 3 main modules: a feature extractor backbone, a feature pyramid network, and 2 detection heads

3.2.1 Backbone Network

We employed ResNet101 [24] as the backbone network to provide rich visual features for following object detection. ResNet stacks 1 convolutional layer with 7×7 convolution (Conv) and 33 residual blocks. Each residual block consists of 3 convolutional layers with different kernels (in 1×1, 3×3, and 1×1 sizes) and different numbers of filters (64, 128, 256, or 512) to generate multi-scale feature maps. All of the generated feature maps are followed by a non-linear ReLu activation [58]. Additionally, each block adds a skip connection from the input signal to output feature maps to combat vanishing gradients.

3.2.2 Feature Pyramid Network

As previously mentioned, a major challenge in object detection is dealing with targets of different size. Fortunately, the residual blocks in ResNet101 naturally provide feature maps at multiple scales and can be used to form a feature pyramid for building semantic feature maps at multiple scales. In RetinaNet, the feature maps from the 2nd, 5th, 27th, and 33rd residual blocks are aligned with their dimensions by up-sampling layers and stacked to build an FPN with lateral connections, as shown in Figure 3.2. To locate objects of different sizes, RetinaNet pre-defines a set of anchors with various heights and widths. With these anchors, object detection tasks are reduced to predicting probabilities and refinements on these anchors. To do this, the feature maps of the anchors were cropped from FPN and sent to detection heads for prediction.

3.2.3 Detection Subnets

The two subnets are designed to separately predict the class and target coordinates from the feature maps of each anchor. The class subnet maps the feature maps of each anchor to a score via four convolution layers with a 3×3 kernel and softmax function. The regression subnet reduces the same feature maps to 4-dimensional coordinates via four separate convolution layers with a 3×3 kernel.

3.2.4 Model Training

To train RetinaNet, we first set the sizes and ratios of height/width of anchors to 0.5, 1, and 2. The pre-trained weights of ResNet101 from ImageNet were loaded into RetinaNet as the initialization for the backbone network. All of the training images were resized to 800*800 before being fed into RetinaNet. IOU (Intersection Over Union) was used to calculate the ratio of intersection and union between annotated objects and anchors. The anchors with an IOU greater than 0.6 over a text region/arrow/T-bar region were labeled as true anchors and were labeled with the corresponding categories. Anchors with an IOU less than 0.4 over any object were considered negative. The anchors with IOUs between 0.4 and 0.6 were ignored to reduce computational overhead.

For the classification subnet, considering the highly unbalanced distribution between easy cases and hard cases, the focal loss [23] was utilized as follows:

$$\mathrm{FL}(\mathrm{p_t}) = -\propto_{\mathrm{t}} (1-\mathrm{p_t})^{\gamma} \log(\mathrm{p_t})$$

$$\mathrm{p_t} = \begin{cases} \mathrm{p} & \text{if } \mathrm{y} = 1 \\ 1-\mathrm{p} & \text{otherwise} \end{cases} \tag{3.1}$$

p is the predictive probability and γ (2 as the default) denotes a regulatory factor used to control the weight of easy samples. For easy cases (i.e. when p_t approaches 1) γ has the effect of scaling the loss toward 0. Conversely, when p_t is near zero (as for uncertain predictions) the scaling factor is closer to 1. By applying γ, we effectively put more weight to ambiguous/hard cases. Alpha simply balances positive and negative samples and was set to 0.1 following the recommendation from [23].

For the regression subnet, we employed a Smooth L1 Loss function [20] as follows:

$$loss(x,y) = \frac{1}{n}\sum_{i=1}^{n}\begin{cases} 0.5 * (y_i - f(x_i))^2\ , & if \mid (y_i - f(x_i) \mid < 1 \\ \mid y_i - f(x_i) \mid - 0.5\ , & otherwise \end{cases} \qquad (3.2)$$

y_i denotes the ground truths and $f(x_i)$ represents the predicted coordinates. This function smooths the gradients when the distance between the predicted coordinates and the ground truth is less than 1. In the other cases, it stabilizes the gradients to avoid gradient explosion. The two subnets were optimized by Nadam [19], with learning rates starting from $5e^{-4}$ and decaying to $1e^{-5}$. Because the arrow and t-bar bodies denote activation and inhibit interactions and their heads indicate the direction of the interaction, we trained two RetinaNets to detect arrows/T-bar bodies and their heads separately.

3.2.5 Model Inference

Once the models have completed training, we can predict the arrow/T-bar bodies and arrow/T-bar heads from any input pathway figures. Typically, the top 1000 bounding boxes with the highest scores from the classification subnet are returned by default with their predicted coordinates. We also set a minimum classification confidence threshold of 0.7 to remove low confidence predictions. Among these remaining bounding boxes, some redundant boxes around the same object still remained. We used a non-maximum suppression operation to filter out duplicate boxes for same object. The IOUs for all of the remaining bounding boxes were calculated and the overlapping boxes with higher IOUs than 0.5 were also treated as duplicates. Among these duplicates, only the bounding box with the highest score was retained. Through such post-processing, we detected all of the arrow/T-bar bodies and arrow/T-bar heads associated with the gene interactions for further processing.

3.3 Gene Name Recognition

3.3.1 OCR Tool

To extract gene name candidates, we leverage Google's Could Vision API. We chose this tool to simplify our pipeline as it can process entire images directly without the need for additional preprocessing (e.g., de-skewing, resizing, etc.). The API response includes all the words found in the figures and a hierarchical breakdown specifying pages, blocks, paragraphs, words, and symbols from the text. By extracting the words in each paragraph, we can build phrases that appear frequently in our figures. The OCR also returns the corresponding bounding boxes and are used to build subsequent gene interactions.

3.3.2 PubTator Central-based OCR Correction

Since misrecognition frequently occurs in the OCR output, corrections from other methods are necessary to improve the OCR results. For this purpose, we used Named Entity Recognition (NER) to improve the gene extraction accuracy. NER identifies and categorizes certain types of entities in text. For our study, we used PubTator Central [8][6][59] to obtain gene annotations from the articles containing pathway figures. PubTator Central is a state-of-the-art NER model that features multiple biological concept annotations and a web API. PubTator Central uses GNormPlus to annotate genes with high accuracy. It achieved an F-score of 86.7% for gene identification on a set of PMC full-text articles and PubMed abstracts. Additionally, PubTator saves the processed data from each abstract and full-text article in a database. In this way, we are able to retrieve gene annotations programmatically through its RESTful API just using an article’s PMC ID. To correct gene outputs from the OCR, we match our results with PubTator Central gene annotations for a given article. To avoid the impact of character cases, we unified all OCR results and PubTator annotations to upper case during matching but left their original symbols for downstream analysis. If no direct match was found, we attempted a fuzzy match using the Levenshtein distance [60] to score unmatched OCR text with all of the gene annotations from PubTator Central. The Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one word into the other. If the score of an OCR recognized gene name was over a certain threshold, set at 90, with 100 being a perfect match, we selected the corresponding gene annotation as the final gene name.

3.4 Gene Interaction Prediction

To extract the gene interactions, we combined several postprocessing methods on the filtered genes, detected indicator heads, and detected indicator bodies. First, we found the corresponding head for each indicator body. This was done by selecting the head bounding box with the largest IOU for each body and calculating its center point marked by the purple dot in Figure 3.3b. We used the body bounding box (seen in Figure 3.3a) to reduce our search space and limit possible noise from intersecting relationships. The head then served as a reference point to find each corresponding tail (i.e., the starting point of each indicator). This was done by finding the detected corner furthest away from each head that was still inside the corresponding body bounding box as marked by the

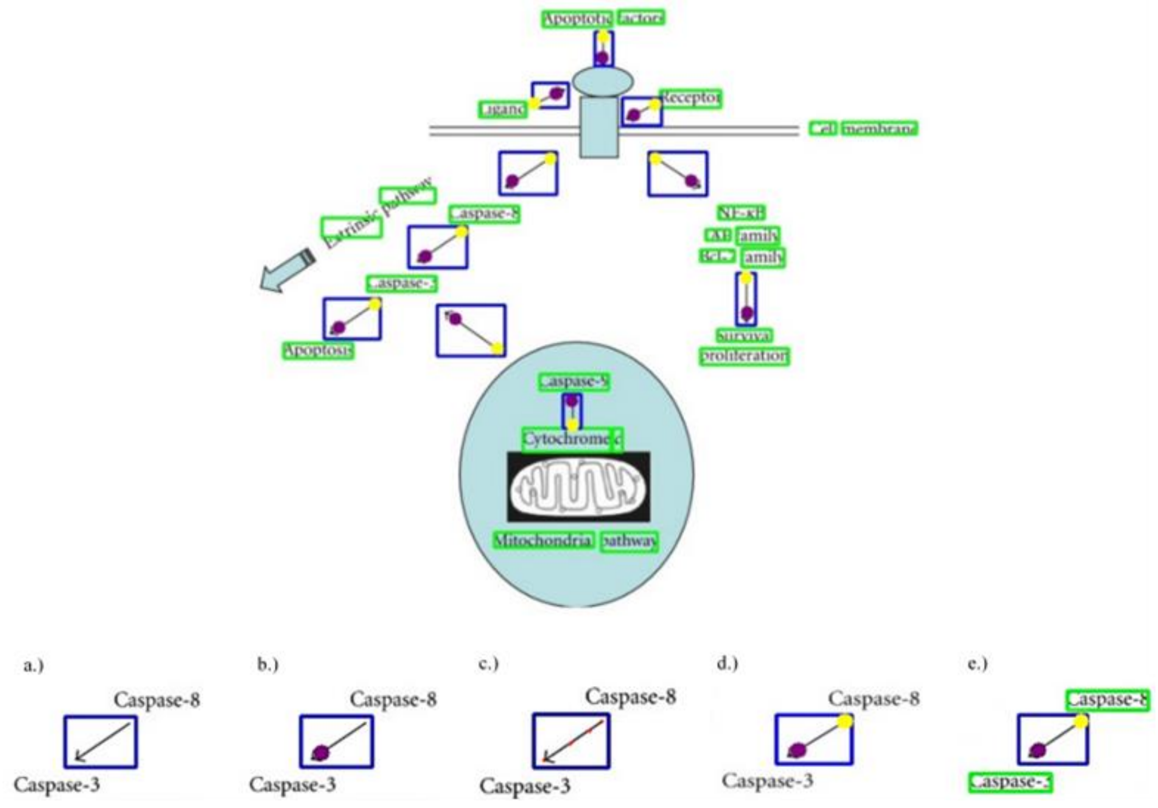
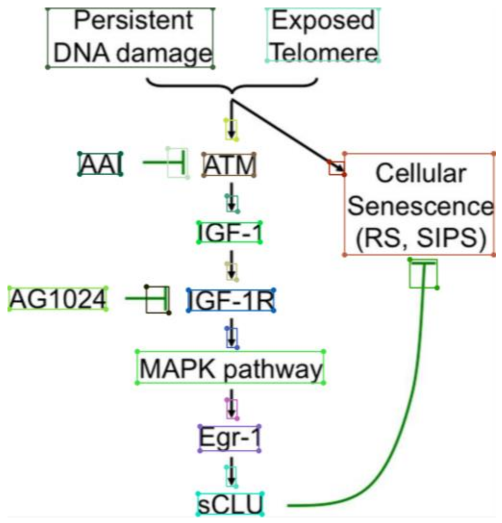


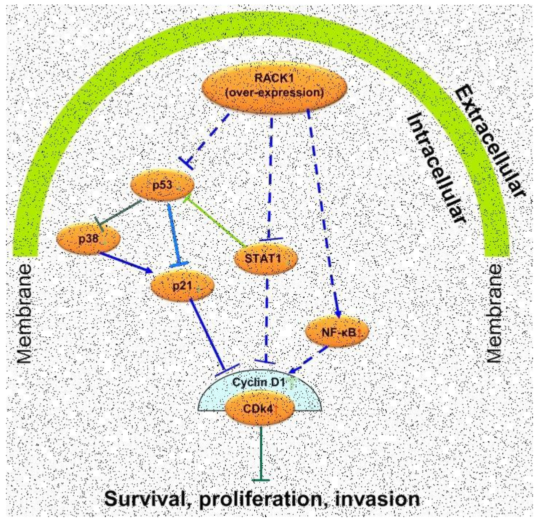
Figure 3.3: Visualization of an example of forming gene interactions with OCR and object detection results.

yellow dot in Figure 3.3d. We used the Shi-Tomasi method for corner detection [61]. An example of the detected corners is shown in Figure 3.3c. Next, using the detected head and tail for each indicator body, we can then find each indicator's corresponding genes. We did this by exclusively selecting the closest entities to the tail and head as our starter and receptor marked by the green boxes in Figure 3.3e. This post-processing schema is demonstrated in our experiments to be an effective way to extract gene interactions from simple pathway figures.

3.5 Data Preparation



a.)



b.)

Figure 3.4: (a) shows an example of a fully annotated pathway figure. (b) shows an example of an augmented sample from our training set.

Due to the lack of a pathway curation benchmark dataset, we constructed a dataset for training our object detection models. 403 pathway figures were directly retrieved from PubMed using pathway-related keywords. Non-pathway figures were manually removed from returned images. These randomly collected pathway figures from different journals and authors were plotted in various styles, which offers sufficient diversity for modeling. To annotate the locations of all arrow/T-bar bodies and arrow/T-bar heads used in the training, we labeled their bounding boxes and categories using the ‘labelme’ tool [62]. To enlarge our data size, we augmented the collected images by applying several transformations to the images. We set random brightness factors from 0.6 to 1 and randomly applied salt & pepper augmentation to quadruple the number of training samples. Upon analyzing our initial dataset, we found inhibit indicators (T-bars) were far less common than activations (arrows). Specifically, in our 403 training images, we found 4,681 arrows but only 931 T-bar lines. To balance the class distribution, we replaced a number of arrows with simulated T-bar lines on the original images after data augmentation. This was done by first randomly erasing a certain proportion of the arrows based on their annotations. Then we plotted a replica T-bar, which followed the original arrowhead position and orientation. In the end, we obtained 4,040 training samples in total, which included 89,740 text regions, 34,127 arrows, and 30,245 T-bar lines.

To evaluate our pipeline, we built a validation set by holding out 45 pathway figures from our training set. In total, this validation set contained 561 genes and 367 gene interactions.

3.6 Results and Discussion

3.6.1 Gene Identification Results

In this section, we report an evaluation of our gene identification using the OCR in terms of recall, precision, and F1. To identify the value of post-processing, we compared the results from the raw OCR output with the post-processed output as well.

Table 3.1: Pipeline performance on gene identification.

Method	Precision/%	Recall/%	F1/%
OCR	53.9	**86.7**	66.5
OCR with post-processing	**75.8**	71.4	**73.5**

As previously described, we can see in Table 3.1 that the Google-OCR output can include non-gene text from the images, typos, and false words caused by artifacts surrounding the target genes. With the gene annotations from PubTator, we can see that our post-processing successfully filtered out many non-gene text results. This greatly increased our precision since we only included real gene names for filtering. However, this filtering seemed to exclude some potential genes as well due to imperfect PubTator annotations. Looking at the F1 score, we find that using our post-processing can achieve a satisfactory trade-off between precision and recall for identifying gene names from pathway figures.

3.6.2 Object Detection Results

We evaluated our trained object detection models on the validation set and determined a correct detection by measuring the overlap between the predicted bounding boxes with the ground truth using 0.5 IOU as the threshold. Based on this criterion, the precision and recall per category are reported in Table 3.2.

Table 3.2: Object Detection models performances on detecting arrows and T-bars

Model	Category	Precision[%]	Recall[%]	F1[%]
Model1	Arrowhead	**94.1**	77.4	84.9
	T-bar head	92.0	**84.0**	**87.8**
Model2	Arrow body	**95.3**	81.3	87.75
	T-bar body	91.1	**88.9**	**89.99**

The first model that detected arrowheads and T-bar heads achieved 77.4% and 81.3% recall, and both more than 90 percent precision on the independent set. Regarding the indicator bodies, the second model performed similarly excellent in recall and precision. Comparing the two types of indicators, the arrows had slightly better performance on both the head detection and shape detection. We observed that some of this error can be attributed to the similarity of T-bars and the letter ‘T’ causing misrecognized T-bars.

3.6.3 Interaction Extraction Results

Table 3.3. Pipeline performance on full gene-interaction recognition

Interaction	Shape	Precision[%]	Recall[%]	F[/%]
Activation	arrow	57.0	75.3	64.88
Inhibit	T-bar	37.4	63.9	47.18
All	N/A	53.7	72.5	61.7

These object detection results laid a solid foundation for subsequent gene-interaction extraction. By combining the indicator locations with the localized text regions, we connect the relationships as previously described in section 3.4. Since the relation results were defined as <starter: relation_type: receptor> semantic triplets, we assigned a true positive when all three components were correct. A prediction with incorrect components or roles was marked a false positive. All missing gene interactions from curations were treated as false negatives.

Under such evaluation we record the recall and precision for the two types of extracted gene interactions on our validation set in Table 3.3. 72.5% of all gene interactions were successfully retrieved from the pathways figures. Among the relationships returned, 53.7% of the gene interactions were entirely correct triplets. Expectedly, some of the incorrectly recognized gene names from previous steps accounted for some of the error. Another source of error could be from our relation prediction depending on just the relative locations between all genes and relation indicators. This could cause mistakes when pairing given several genes clustered around the same relationship indicator. Between the two types of gene interactions, the activation relationship again was extracted more accurately than the inhibiting interaction. Since activations are the prominent type of gene interaction in pathway figures, our pipeline still shows use as an automatic curation approach by supplementing rich gene interactions for downstream analysis.

3.7 Summary

In this section I introduced a framework that integrates two different modes for extracting relationship triplets. We demonstrated that the pipeline is able to extract genes and their relationships successfully on a held-out portion of the training data. However, we observed that our object detection models struggled to generalize to new data. In the next chapter, I introduce a rule-based augmentation method for document and diagram data to bridge this gap.

Chapter 4

Synthetic Data Generation

In this chapter, I introduce a novel data augmentation method for up-sampling figure/diagram data. While the use-case explored was for pathway figures, it can be applied to different domains as well.

4.1 Overview

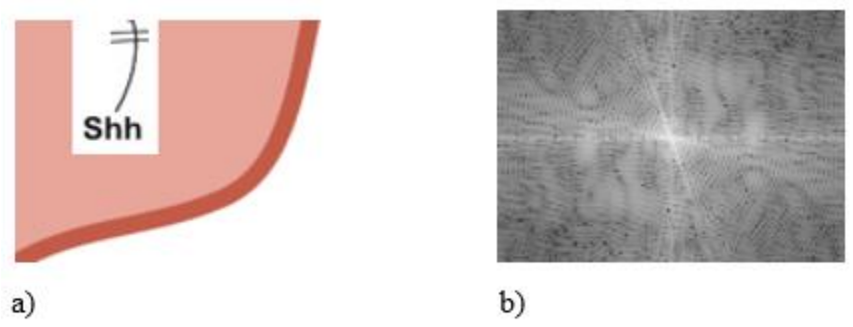


Figure 4.1: Filtering candidate regions for high-frequency components. (a) a sample slice of a pathway image; (b) radial profile of the slice in the spectral domain.

In the previous chapter, I introduced a unified pipeline for extracting gene interactions from articles. While this pipeline performed very well on a validation set with samples from a similar distribution as the training set, we observed that it failed to generalize well to new data during testing. This was likely because of too little training data leading to a model with high variance. While we did apply several forms of data augmentation, it was observed that these traditional forms of augmentation provided diminishing returns after enough samples. This is likely because simple augmentations can increase the total number of training images several times over but did not increase the number of unique relationships seen during training. To address this shortcoming, I introduced an approach that generates fully synthetic samples. Since many relationships in pathway diagrams follow a simple structure, they are easy to reproduce. To review, there are just a few components that make up a relationship in a pathway figure: two entities and a connecting identifier. These identifiers can be represented as arrows, objects, or by proximity. Here we will again first target the two types of indicators that are most frequently encountered in our pathway figures: arrows (activate) and t-bars (inhibit). We generate our fully synthetic relationships by first identifying an empty region on a template image. As we generate new images, we cannot place the slices randomly on the templates, as there is a large potential for overlap that does not exist in the ground truth figures. So, we first sample a random candidate region on a

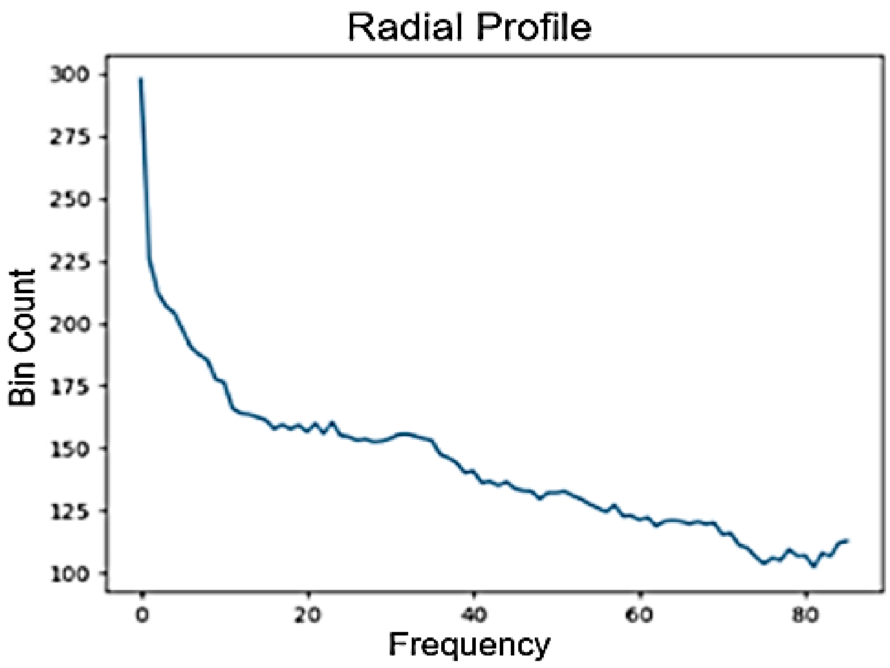


Figure 4.2: Radial profile for the candidate region in the spectral domain from Figure 4.1. The x-axis shows the pixel radius from the center as our frequency analog. The y-axis represents the number of white pixels at each radius.

template (Figure 4.1a) where our relationship could be placed. Then, we convert that destination region to the spectral domain (Figure 4.1b) via a fast-Fourier transform [63] and calculate the radial profile of that slice in the spectral domain (Figure 4.2). The radial profile is the sum over pixel values the same radius away from the center of the slice. We do this to see how many high-frequency components exist in that destination region since the increased radius corresponds to higher frequencies in the source image. In images, high-frequency components correspond to edges, contrast, and complex shapes. Our intuition is that pre-placed slices on the image will then be represented in the high-frequency regions. By setting a threshold on those high-frequency components, we can effectively search for good placements on the templates.

With a region selected, we then determine our entity placement given the area’s dimensions (see Figure 4.3a). Next, we draw a spline between the two textboxes (Figure 4.3b) and add an indicator head at one of the spline ends (Figure 4.3c). This class identifier can be an arrowhead or a t-bar (for activate or inhibit relationships). This approach effectively mimics the structure of many pathway relationships. Using such fully synthetic data generation has several advantages over traditional augmentations for this task. For instance, the model can train on a more diverse set of training data since there are no repeated relationships. This enables us to target specific types of exotic relationships that were previously more difficult to categorize due to little data: (e.g., curvy arrows, splines with corners, or dashed splines). Additionally, this process can be multi-threaded to generate many diagrams at once.

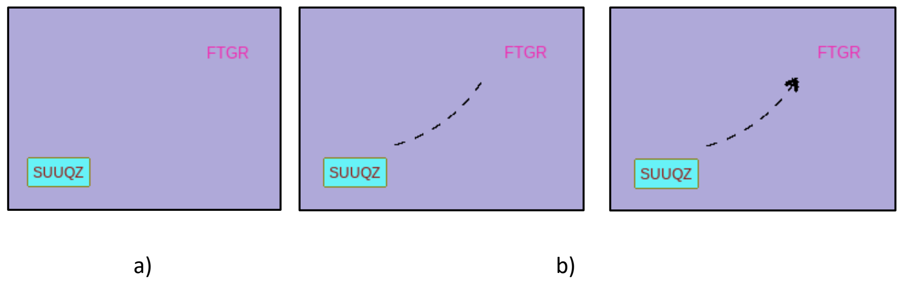


Figure 4.3: To generate a relationship, with two placed entities (a), we denote their relationship by drawing a spline between them (b) and placing an identifying indicator at one end (c).

4.2 Implementation Details

4.2.1 Checking Background

While simple to outline, implementing each step of this process is more involved. In the case that all pixels in the destination region are the same, we do not have to run the full spectral check and can immediately stitch a relationship. However, when this is not the case, we convert the destination region to the spectral domain and generate its radial profile as previously mentioned. Notably, when calculating the radial profile on this output, we must normalize by distance to the center of the region since as the radius grows so does the number of pixels at that radius. To filter out regions with too many high-frequency components, we look at the binned statistics over the radial profile. Specifically, we bin the radial profile into 4 sections and look at the 2 later bins corresponding to the higher frequencies. If the binned mean for either of those regions is too large, then we can rule out this placement. We used a threshold of 50 to filter out slices with too many high-frequency components to determine our placement. This specific threshold balances allowing color gradients while still removing any slices with harsh edges. We found this method to be more effective than simply looking at the pixel statistics of the destination slice and setting a threshold for the standard deviation. This approach failed on edges of similar pixel values. Using our method produced more realistic figures that better resembled the source dataset.

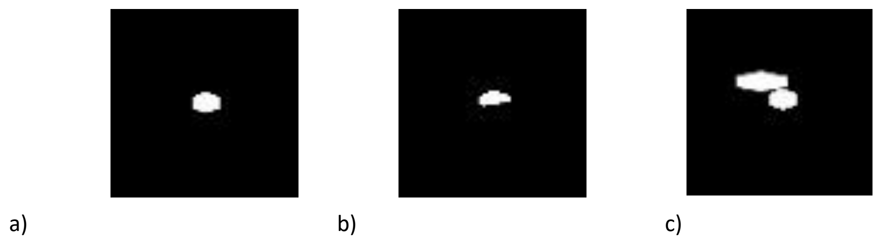


Figure 4.4: To generate a cluster of entities, we started from the shape masks of two entities and iteratively moved one shape's center until the IoU between the shapes was 0.

4.2.2 Entity/Cluster Generation

The textboxes of pathway entities come in a variety of shapes. To mimic this variety, we pulled from a folder containing images of arbitrary shapes. These shapes are extracted from the source images and transformed to fit the dimensions of our text. To further generate clusters of entities, we start from one shape and its mask (Figure 4.4a). To add another shape to the arrangement, we select a random direction from the first entity's center and set the center of the new shape as some distance along this path. The initial distance is a factor of the first shape's dimensions. We then calculate the intersection-over-union between the two shape's masks (Figure 4.4b). If there is any overlap, we then increase the push factor along our selected direction and repeat until they are non-overlapping. This process can be repeated to add any number of entities to each cluster.

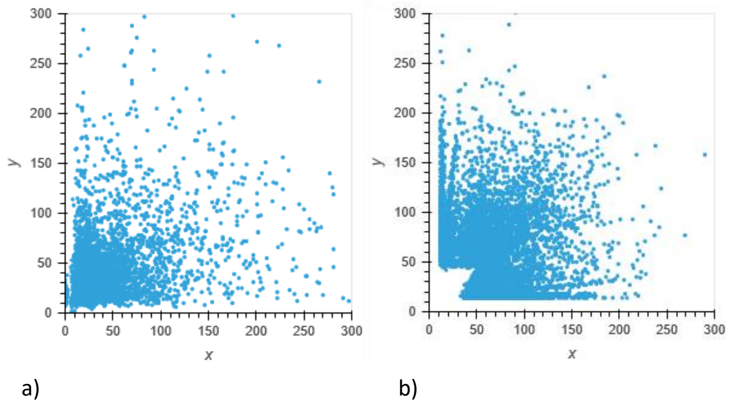


Figure 4.5: Histograms showing the distribution of relationship sizes in real pathways (a) and our synthetic pathways (b).

4.2.3 Entity Placement

To effectively replicate the entity positions seen in real pathway diagrams, we look at the histogram of relationship dimensions seen in those figures. Figure 4.5a shows a 2D-histogram of 1000 diagrams which highlights that those relationships are most concentrated in the low dimensional regimes. We use this distribution to guide the region selection of our algorithm and fix the position of our two entities to opposite corners. We do this to let the real relationship's dimensional distribution fully guide our entity placement. However, we do maintain that the region selected for placement must be large enough to contain both entities. This explains the gap in Fig 4.5b that is not seen in the real data. In the case of extreme dimensions (e.g., much larger x than y and vice versa), we fix the placement to be top-down or left-right. We do this since most relationships follow this orientation.

4.2.4 Drawing Spline

Once the entities have been placed, we can use their centers as reference points for the connecting spline. We use three different types of splines: lines, arches, and corners. For direct lines, we must first find the start and endpoints for the spline by interpolating a direct line between the centers of the two textboxes. We then select the n-th point along the line outside the textboxes as the respective start and end points. We set n dynamically based on the distance between the two textboxes. With the start and end points, we then draw a line between these two anchor points as our spline. If we are drawing an arch, we start with the same method for obtaining the start and end points. Then, we calculate the slope perpendicular to the line between them. With this, we can calculate a third anchor point which will mark the apex of the arch. We set a parameter to determine how far from the baseline the third reference point should be. This allows us to control how ‘curvy’ each arch is. Using all three anchor points, we can then interpolate a spline between them. When drawing a cornered spline, we use a different method for obtaining the start and end points. For each square configuration, we can have two different placements for the start and end points. They can be placed outside the textboxes at some pre-set distance towards the same empty corner. Then we use their max or min dimensions to determine a corner point for the third anchor and connect these three points. To draw dashed splines instead, we simply omit placing intervals of the spline. These intervals are drawn from random bounds and are dependent on the thickness of the spline.

4.2.5 Drawing Indicator

With the spline drawn, we now place an indicator head at one of the ends. Since there are different styles of indicator as well, we follow the same approach as with entity shapes and pull from a set of indicator shape images. Again, we extract and transform the indicators as needed for the given spline style. We also rotate these indicators to follow the slope of the spline near the end point and place that transformed indicator onto the end of the spline.

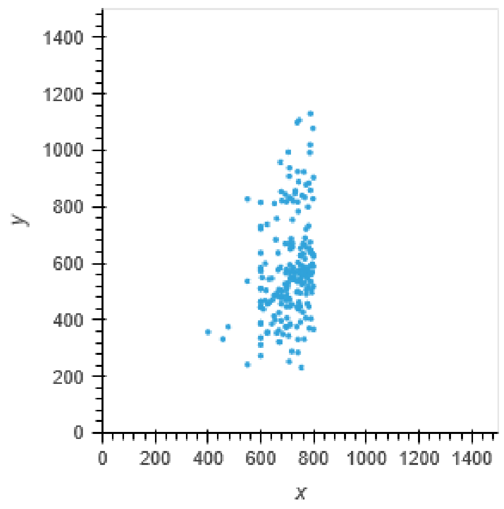


Figure 4.6: Histogram showing the distribution of pathway image dimensions.

4.2.6 Parameter Configuration

The above methods detail how to generate a new relationship, but we can control many of the parameters for this process to ensure that each one is sufficiently unique. For each label, we can control the font color, style, size, and thickness. For each textbox, we can control the textbox margin, background color, textbox shape, and border thickness. For each spline, we can control the indicator placement, type, length, width, color, and thickness. We dynamically change these parameters during generation. This enables us to generate a more robust dataset that mimics the wide variety of relationships that are seen in the real data. For instance, we use the image dimension distribution from real pathways (Figure 4.6) to set the bounds for our templates.

4.3 Experimental Setting

4.3.1 Model

For our relationship localization experiments, we again evaluated our method on the widely used RetinaNet [23] architecture with a ResNet-50 [24] backbone pre-trained on the ImageNet [64] dataset. In all of our experiments, we finetuned this model for 50 epochs with a learning rate of 0.01. We are limited to 1 image per batch due to memory constraints. For loss, we used a combination of the sigmoid focal loss [24] for classification (for imbalanced class distributions) and dense box regression for localization.

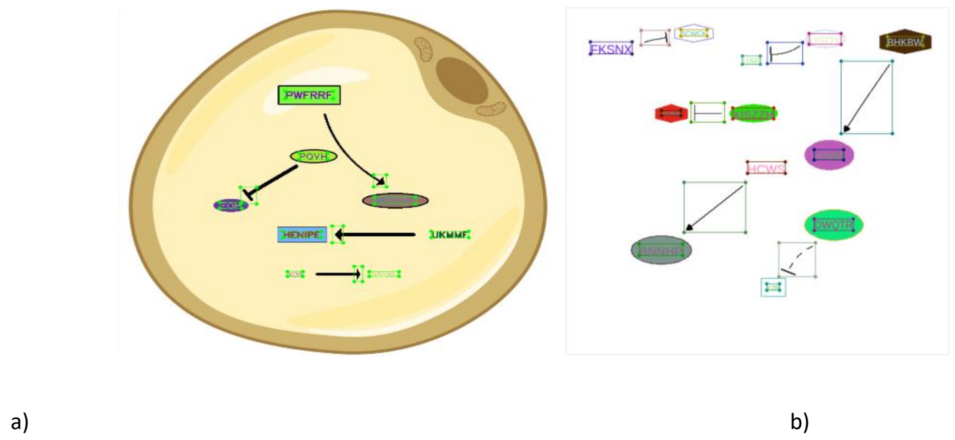


Figure 4.7: Synthetic samples with annotation. (a) shows the indicator head annotations used in Experiments 4.4.1 & 4.4.2. (b) shows the indicator body annotations used in Experiment 4.4.3.

4.3.2 Data

For the base augmented dataset, we use the same dataset described in Chapter 3. This dataset uses salt & pepper, color correction, and random noise to generate 4,000 training images. We treated training with the augmented data alone as our benchmark. For the synthetic data, we generated three sets of increasing size with 1,000, 3,000, and 6,000 images (Figure 4.7a as an example). We then looked at how different amounts of synthetic data coupled with the augmented dataset can improve the generalization of our model. We evaluated with new 45 images collected from PubMed as our validation set and report the mean-average-precision (mAP) over the three classes: inhibit indicators, activate indicators, and gene text. The mAP measurement captures how well all objects are detected and classified. Results displayed in the tables are averages and standard deviations over three runs.

4.4 Results

4.4.1 Synthetic Data for Mixed-Batches

Table 4.1: Testing mAP for increasing amounts of synthetic data used.

Training Data	Aug	Aug + 1k Syn	Aug + 3k Syn	Aug + 6k Syn
mAP	26.9 +/- 2.9	26.5 +/- 0.4	28.8 +/ - 2.4	**29.1 +/ - 2.1**

In this set of experiments, we look at how our method can affect the generalization of the model (shown in Table 4.1). With the 4000 augmented samples as our baseline, we see how increasing the amount of synthetic data used in mixed batches affects validation. We find that our method used in conjunction with the augmented data improves the generalization of RetinaNet. This is likely because we can introduce unique relationships/features that simple augmentations cannot. This notion is supported by the fact that increasing the amount of synthetic data generated by our method continues to improve the performance of the model.

4.4.2 When to Use Synthetic Data

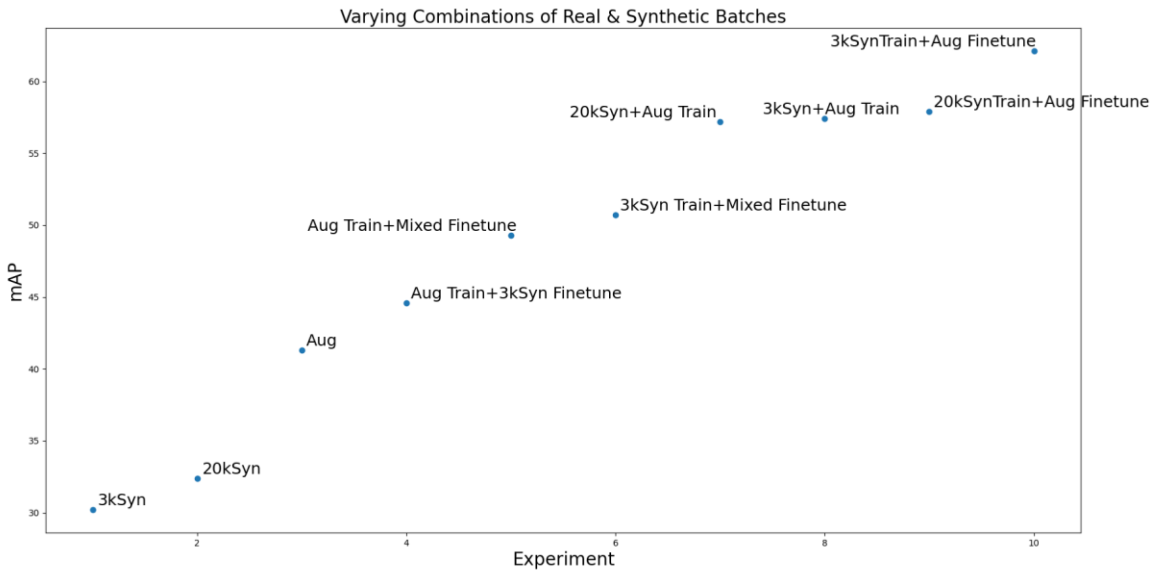


Figure 4.8: Comparing combinations of real and synthetic data at different stages of training.

Following the improvement shown by leveraging synthetic data in mixed batches, we also sought to understand how different combinations of real and synthetic data affect training. To that end, we first measured the performance of the augmented data and different amounts of synthetic data on the independent set of 45 images using mAP:.50. As seen in Figure 4.8, increasing the amount of synthetic data, as expected, showed improvement in generalization from 3k at 30.2 to 20k at 32.4 mAP. However, both were unable to fully close the gap to the augmented data 41.3 mAP. Interestingly, if we first pretrain on the augmented data with a learning rate of 0.01 and finetune with synthetic data at 0.005, we also see improvement over augmented training alone at 44.6 mAP. This improvement is even more pronounced if we include the augmented data in our finetune stage and use mixed batches (reaching 49.3). We flipped this experiment to pretrain with synthetic and finetune with mixed batches and can see another boost in performance (50.7 mAP for 3K synthetic). Although, as previously validated, the augmented and synthetic data seem to capture different features and when used in combination improve over either standalone performance. In the case that we trained with mixed batches from the start (our setting from experiment 1), we reached 57.2 and 57.4 mAP for 20k and 3k synthetic samples. However, our best setting came from pretraining on synthetic and finetuning on real data reaching 62.1 mAP.

4.4.3 Generalizing to New Tasks

Table 4.2: Testing mAP for increasing amounts of synthetic data used from each method, starting with the base augmented dataset.

Training Data	Syn	Aug	10k Syn + Aug	10k Syn -> Aug
mAP:.50	29.1 +/- 2.8	52.9 +/- 1.8	63.3 +/- 3.0	**66.3 +/- 3.9**

For our third experiment, we look to test how well our up-sampling approach improves generalization for additional and more complex classes. To that end, we tried to localize the entire relationship indicator bodies and specify two additional gene-gene relationship markers (indirect activate and indirect inhibit). These classes are differentiated from their bases with dashed bodies as seen in Fig 4.7b.

We annotated the same 250 real diagrams used for the relationship heads, but instead augmented their bodies to 3000 samples. As shown in Table 4.2, training with mixed batches again shows considerable improvement over either alone. We also leveraged the insights from experiment 2 and we pretrained on 10,000 synthetic samples with annotated bodies and finetune on the augmented real samples. This approach led to a 25% improvement over the model that was just trained with augmented samples.

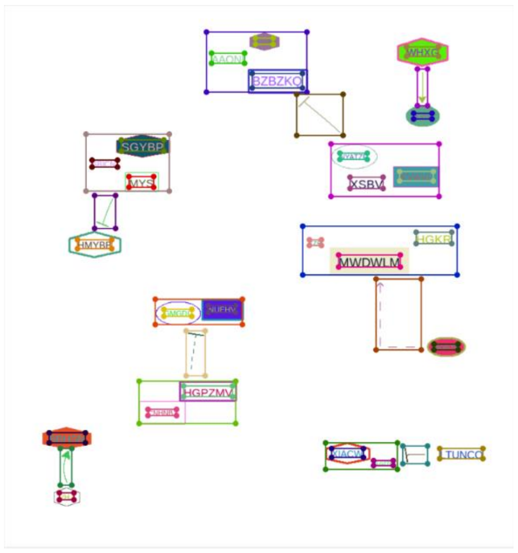


Figure 4.9: An example of a synthetic cluster sample with annotation.

Table 4.3: Testing mAP for RetinaNet trained on Synthetic Clusters.

Dataset	# Genes	# Clusters	mAP:50	mAR:50
Real Testing	616	64	49.3	79
Syn Validation	7259	2303	83.4	92.7

To further test how well our method can generalize to new target classes, I extend training to a different type of identifier. In these experiments, I target localizing clusters. These clusters are identifiable by proximity between entities instead of by any specific shape. For this experiment, I generated 12,000 samples using the previously described methodology (Figure 4.9), where 2000 are held out for validation. A small manually annotated set of 25 images is also used for testing on real data. Following the previous experiments, I trained for 50 epochs and saved the checkpoint with the best validation performance. Looking at the validation performance from Table 4.3, one can see that synthetic clusters are learnable training signals for the model. This generalization extends to real data as well and reaches decent average recall and average precision on our testing set. This further validates our method's generalizability to new and more complex target classes.

4.5 Introducing Noise

$$\mathbf{SNR} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} |f(x,y) - \hat{f}(x,y)|}{M * N} \qquad (4.1)$$

Following the success of jointly training with synthetic data, we looked to see if we could improve upon the baseline performance of the models trained on solely synthetic data. To that end, I experimented with incorporating different amounts and combinations of structured and unstructured noise during generation. My motivation was that many real pathway diagrams contained artifacts and background geometries that were not robustly represented in the synthetic data. Additionally, learning to become noise agnostic can promote focus on underlying target signals and make our model be more context aware. Fortunately, measuring noise is a previously studied subject across signal and image processing. For my experiments, I used the average pixel difference (4.1) to measure noise, since the conventional signal-to-noise ratio measurement assumes uniformity from the noise distribution. I also recorded the average precision and recall for each noise-type.

4.5.1 Structured Noise

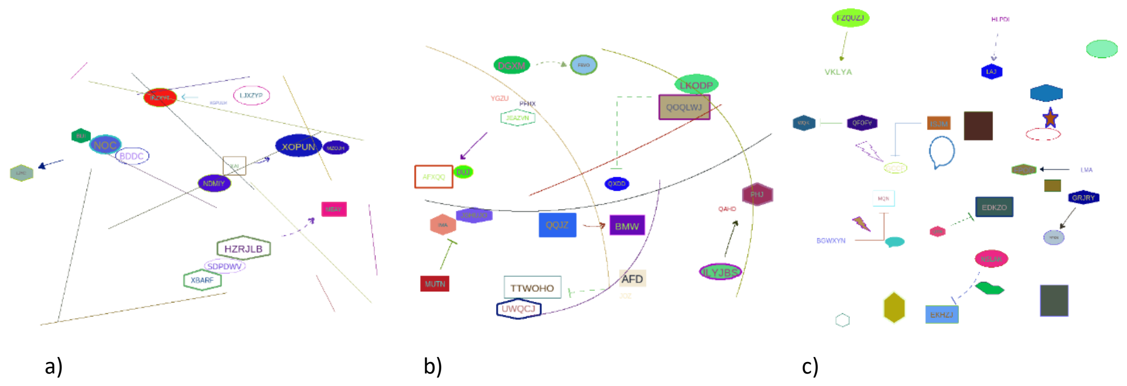


Figure 4.10 Structured noise examples for lines (a), arches (b), and shapes (c).

Table 4.4. Testing performance of purely synthetically trained models using different types of structured noise.

	mAP:50	mAR:50	SNR
Baseline	29.4	75.7	-
Lines (0-10)	38.9	**85.6**	0.002 +/- 0.001
Lines (0-20)	43.7	82.6	0.004 +/- 0.003
Lines (0-40)	38	83	0.008 +/- 0.005
Arches (0-10)	44.8	78.5	0.002 +/- 0.002
Arches (0-20)	45.8	83.5	0.005 +/- 0.003
Arches (0-40)	**46.1**	80.8	0.01 +/- 0.08
Shapes (0-10)	33.6	69.5	0.007 +/- 0.006
Shapes (0-20)	42.9	69.7	0.015 +/- 0.01
Shapes (0-40)	31.1	67.2	0.03 +/- 0.02

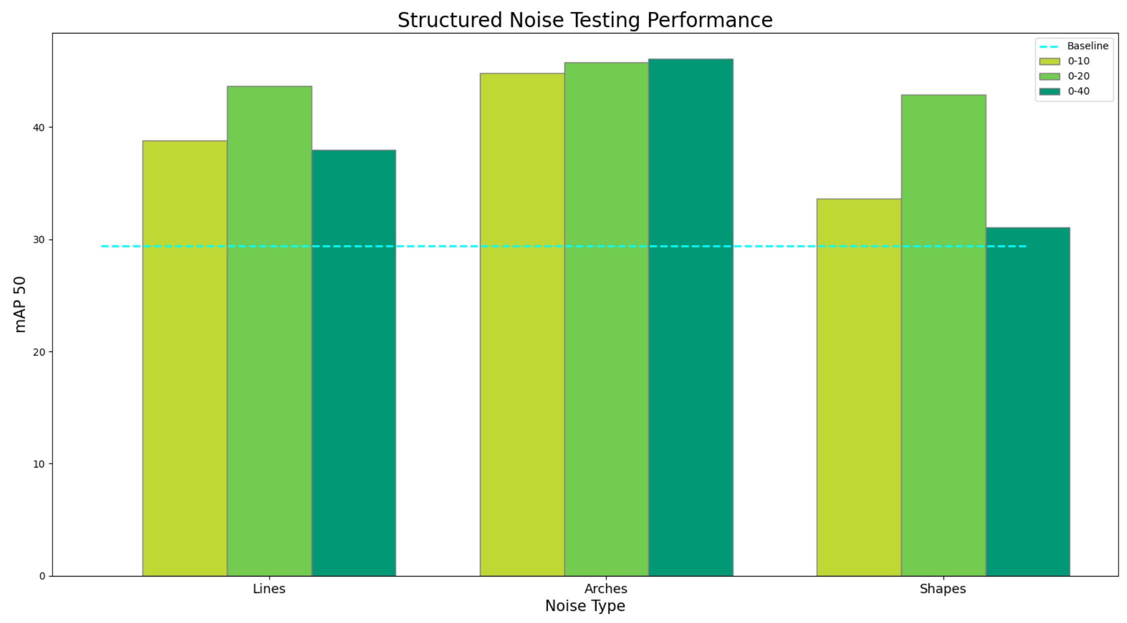


Figure 4.11: Testing performance of synthetically trained models with different types of structured noise (lines, arches, and shapes).

First, I experimented with different types of structured noise. To do this, I introduced varying amounts of lines, arches, and shapes randomly in the generated synthetic images (Figure 4.10). Specifically, I incorporated 0 to 10, 20, and 40 artifacts for each image. To place the lines and arches, I random selected endpoints on the images and connected them. For the shapes, I placed them using the same region selection procedure outlined in section 4.2. As seen in Figure 4.11 and Table 4.4, I found that increasing the number of artifacts and relative noise levels increased the mean average precision for all structured noise types up to a certain degree. However, at 40 maximum artifacts, the performance began to degrade. We can also see that increasing the number of artifacts also increased the relative amount of noise as intended. This highlights the importance of targeted structural noise being represented in the synthetic samples.

4.5.2 Unstructured Noise

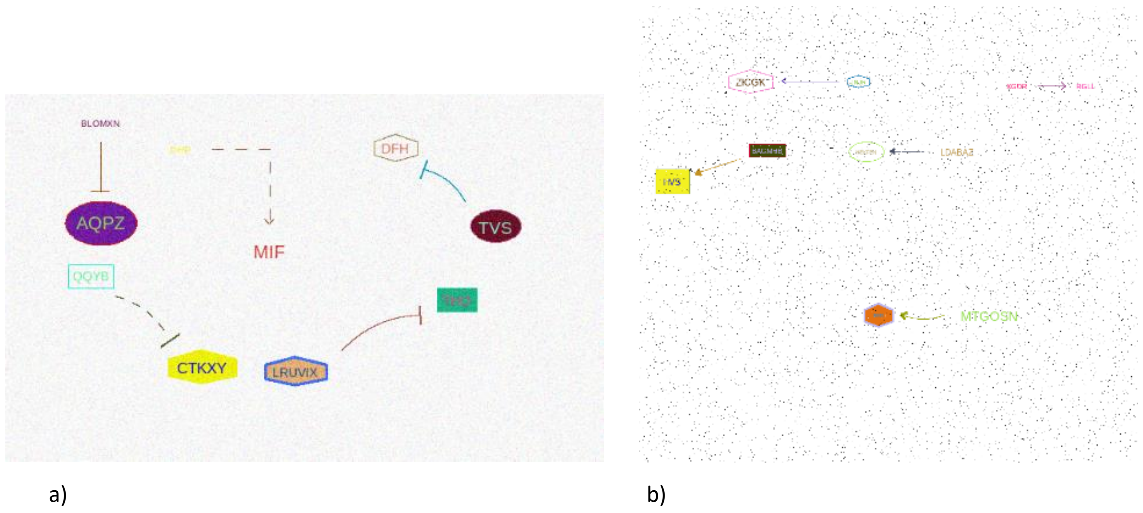


Figure 4.12: Unstructured noise examples for gaussian noise (a) vs. salt and pepper noise (b).

Table 4.5: Testing performance of purely synthetically trained models using different types of unstructured noise.

	mAP:50	mAR:50	SNR
Baseline	29.4	75.7	-
Gaussian (0-150)	28.4	77.7	0.026 +/- 0.008
Gaussian (0-300)	34.7	**79.8**	0.036 +/- 0.013
S&P (0.004)	31.7	65.6	0.006 +/- 0
S&P (0.008)	**35.3**	70.3	0.012 +/- 0

Complimentary, I also looked at how incorporating traditional unstructured noise could affect generalization. For these experiments, I included varying amounts of gaussian noise as a 0-150 and 0-300 pixel value change per pixel (Figure 4.12a). For salt and pepper, I set salt and pepper’s base probabilities as 0.004 and 0.008 (Figure 4.12b). In Table 4.5 I observed that introducing lots of gaussian noise improved over the baseline mean average precision and mean average recall. Adding salt and pepper improved precision, but surprisingly hurt recall compared to the baseline. This may be because too much occlusion over the target signals.

4.5.3 Mixing Noise

Table 4.6: Testing performance of purely synthetically trained models using mixtures of structured and unstructured noise.

	mAP:50	mAR:50	SNR
Baseline	29.4	75.7	
Lines (0-20)	43.7	82.6	0.004 +/- 0.003
Lines (0-20) + Gauss (0-300)	45.2	81.2	0.04 +/- 0.01
Arches (0-20)	45.8	**83.5**	0.01 +/- 0.003
Arches (0-20) + Gauss (0-300)	**46.7**	80.1	0.04 +/- 0.01
Shapes (0-20)	42.9	69.7	0.02 +/- 0.01
Shapes (0-20) + Gauss (0-300)	35.3	67.3	0.05 +/- 0.02

Finally, we looked at mixing noise modalities. We combine the 0 to 20 artifacts from structured noise and gaussian noise for every sample. From Table 4.6, we find that mixing the two reduces recall slightly across the board but does boost precision for all cases except the shapes. Mixing the shapes and gaussian noise may introduce too much noise for the model to discern the target signal.

4.5.4 How Much Noise?

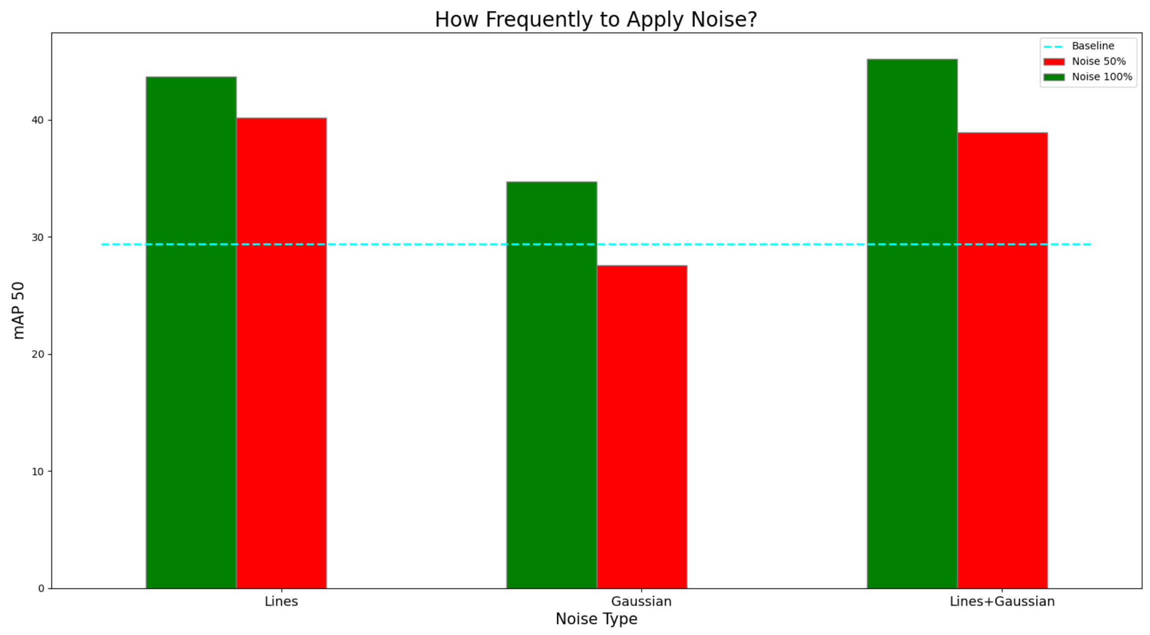


Figure 4.13: Bar-plot showing the effect of halving the amount of different noises in training samples

With a better understanding of the effect of integrating noise with every image, we also looked at the effect of applying this noise on every other image. To test this, I set the probability of including a given type of noise to be 0.5 and reran our experiments for random lines, gaussian noise, and using both. I found a consistent fall in performance by reducing the number of samples with noise (Figure 4.13). Gaussian noise even falls below the baseline. This is likely because only including the gaussian noise in half of the samples does not promote the same noise agnostic behavior that helps the model generalize.

4.6 Summary

This chapter introduced an up-sampling method for object detection on pathway figures that is based on a set of rules and biological domain knowledge. Such biology-inspired data augmentation is a better alternative for up sampling pathway diagrams, since the relationships in these figures are highly diverse and traditional methods for positional or color modification cannot robustly mimic these features. Additionally, GAN-based approaches may not follow the underlying biological meanings. As demonstrated, our method's fully synthetic approach was able to increase the generalization capacity of the transfer-learned models on several tasks. We also validate the value of a targeted up-sampling approach in addition to traditional augmentation and characterized the importance of noise representation in our synthetic pathway diagrams. This work motivates further investigation into the upper bound of this synthetic approach and its possible extensions.

Chapter 5

Active Learning

In this chapter, I introduce a new contribution in the field of active learning. By combining my synthetic data generation approach into an active learning schema, I can train with no data pool or acquisition function. I show how this approach can reduce the training time of our models without sacrificing generalization.

5.1 Overview

With improved standalone performance, I looked more confidently look towards applying synthetic data generation in an active learning setting. This improved generalization was needed, to ensure that the changes made to input selection would correspond with improved generalization. We further explore this dynamic in this chapter. As mentioned previously, how to best leverage data augmentation in an active learning setting is an open research question. My contribution in this section is Pool-less Active Learning (PAL) via synthetic data generation. Previous augmentation methods do not have the fine-grain control necessary to directly leverage a training sample's uncertainty to produce new data following that feedback. As such, all of the methods previously described must estimate what the model may perform poorly on and sample from a data pool accordingly. Whereas PAL can directly take information on what the model is performing bad on to create a sample the model will likely learn more from. I show in my experiments, PAL can reduce the number of iterations needed for training.

5.2 Combining Synthetic Data & Active Learning

As I mentioned, there are no clear answers for how to best combine active learning and data augmentation. Combining active learning with synthetic data is equally underexplored. As mentioned, most previous methods utilize a large pool of unlabeled data to draw from. However, if we have the ability to generate each batch with whichever classes we need, then we don't really need a data pool. Instead, to balance uncertainty and diversity sampling, we could set class probabilities for how often each class should occur as $\{p_{1,}p_{2}...,p_{n}\}$ and generate synthetic samples following these probabilities, such that class n has p_n probability of being represented in our sample. Then, the active learning formulation just reduces to finding how to best update these class probabilities during training. This is our Pool-less Active Learning (PAL) training schema.

To update these class probabilities, we need to consider how the model should learn from data. There are two clear options to consider. The first recommends that a model should see all of the classes it was bad at regardless of what the target class was. The second idea says that if the class prediction for a bounding box is bad, then the model should just see that box's target class more often.

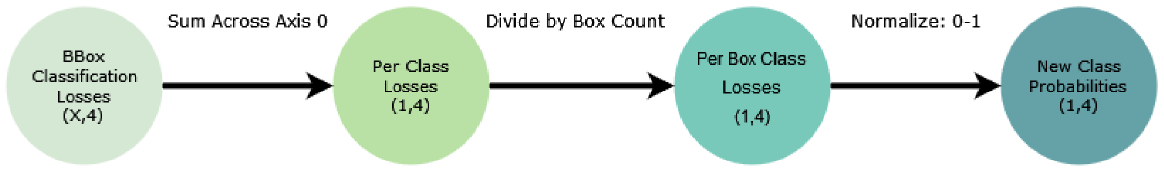

BBox Classification Losses (X,4)
Sum Across Axis 0
Per Class Losses (1,4)
Divide by Box Count
Per Box Class Losses (1,4)
Normalize: 0-1
New Class Probabilities (1,4)

Figure 5.1: PAL 1 implementation for obtaining updated class probabilities from classification losses.

We can implement the first way (PAL 1) as follows (Figure 5.1). From the binary cross entropy losses from all bounding boxes, we first sum across all of the samples. Then, we divide by the total number of boxes and just normalize this value between 0-1 to get our seed probabilities for the next batch. This approach encourages more discrimination between classes and includes uncertainty from background regions. Unfortunately, this approach also has some drawbacks. If a class doesn't occur frequently by chance, it's probability could be reinforced to be low and almost never show up to be learned from.

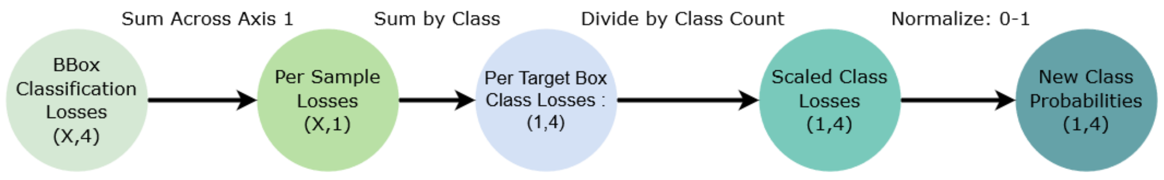

Sum Across Axis 1
Sum by Class
Divide by Class Count
Normalize: 0-1
BBox Classification Losses (X,4)
Per Sample Losses (X,1)
Per Target Box Class Losses : (1,4)
Scaled Class Losses (1,4)
New Class Probabilities (1,4)

Figure 5.2: PAL 2 implementation for obtaining updated class probabilities from classification losses.

The second way (PAL 2) would be implemented as follows (Figure 5.2). From the binary cross entropy losses from all bounding boxes, we sum across each sample to get the classification loss for all of the samples. Then, we add all of those losses together by target class and scale the sums by how many target boxes there were actually for each class. For example, if 45 of the boxes were inhibit, then the inhibit loss would be divided by 45. We normalize scaled losses between 0-1 to give our seed probabilities for the next batch. However, this approach has several problems as well. Just training on the target class more frequently, may not robustly help discrimination between other classes. Additionally, this approach ignores the loss from background boxes.

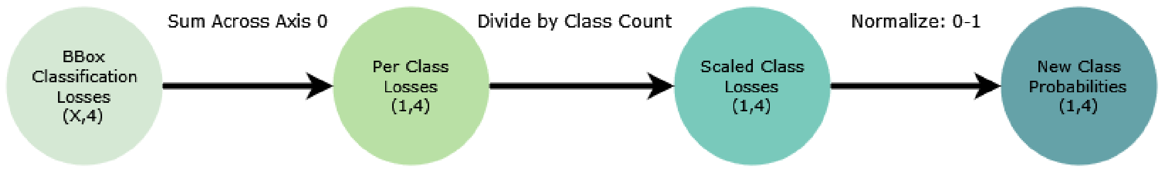

BBox Classification Losses (X,4)
Sum Across Axis 0
Per Class Losses (1,4)
Divide by Class Count
Scaled Class Losses (1,4)
Normalize: 0-1
New Class Probabilities (1,4)

Figure 5.3: PAL Mix design which combines methods 1 & 2 for obtaining updated class probabilities from classification losses.

To balance discrimination between classes and promote diversity sampling, we also experiment with combining methods 1 and 2 into PAL Mix. From the binary cross entropy losses from all bounding boxes, we can sum across all samples (similar to method 2). But instead of scaling by the box count, we scale by the class count (similar to method 1) and then normalize between 0 and 1. Figure 5.3 shows an overview of this process. Ideally, this approach could balance the issues of both while still maintaining their benefits.

5.3 Experiments

5.3.1 Validating PAL

For my first experiment, I want to validate Pool-less Active Learning as an active learning schema. To test this, I compare the testing performance of RetinaNets trained using PAL with standard training and a conventional form of active learning using lowest-confidence predictions. For each training setting, I trained 10

RetinaNet models from ImageNet backbone weights. The PAL models were trained for 15,000 iterations where each iteration was trained on a batch of data that was generated from the classification losses of the previous iteration as previously described. At the beginning, we set the default probabilities to be equal. For the standard training, I learned for the same number of iterations, but all of the samples are generated with an equal probability for each class occurring. Training via lowest-confidence active learning is different. To follow standard convention, I first generated a large synthetic data pool of 50,000 images to pull from. I then initially randomly sampled 1,000 of these images as the initial training set. Next, I trained on this set for 10 epochs and evaluated on the held-out data pool at the end to extract the samples with the least-confident predictions. I added the 1,000 samples with the least confident predictions to the training set and trained a new model from scratch. Then, I repeated this training and sampling cycle until the training set had 15,000 images. Finally, as a fair comparison to the other two methods, I trained a new model from scratch using this dataset for one epoch. I repeated all of these experiments 10 times to obtain a more accurate measure of their performance in relation to one another and reported the mean and standard deviation of their performances on the testing set of 45 images used in chapter 4.

Table 5.1: Testing Performance of PAL trained models in terms of precision, recall, and average number of iterations to model convergence.

Method	mAP	mAR	Avg. Iterations
Standard Training	42.7 +/- 1.3	81.7 +/- 5.3	9860 +/- 1564
PAL 1	0.3 +/- 0.4	0	8575 +/- 1233
PAL 2	41.7 +/- 1.2	**84.5 +/- 1.4**	**8580 +/- 1802**
PAL Mix	41.8 +/- 1.6	83.5 +/- 3.0	8900 +/- 1790
Least Confidence	**48 +/- 1.3**	74.2 +/- 2.6	9000 +/- 1446

From Table 5.1, we can see that PAL 2 achieved similar testing performance to standard training in terms of mAP and mAR. In fact, PAL 2 was able to achieve even slightly higher recall. Very notably, PAL 2 was also able to reduce the total number of training iterations required to reach this generalization by ~13% compared to standard training. Meanwhile, least confidence active learning was able to achieve the highest precision, but its recall suffered as a result. Notably, least confidence active learning also required much longer to train compared to standard training and PAL. This was because traditional active learning requires multiple cycles of training to gradually build up a diverse dataset. In this way, one can see that PAL offers a more balanced approach to active learning in terms of recall, precision, and total runtime.

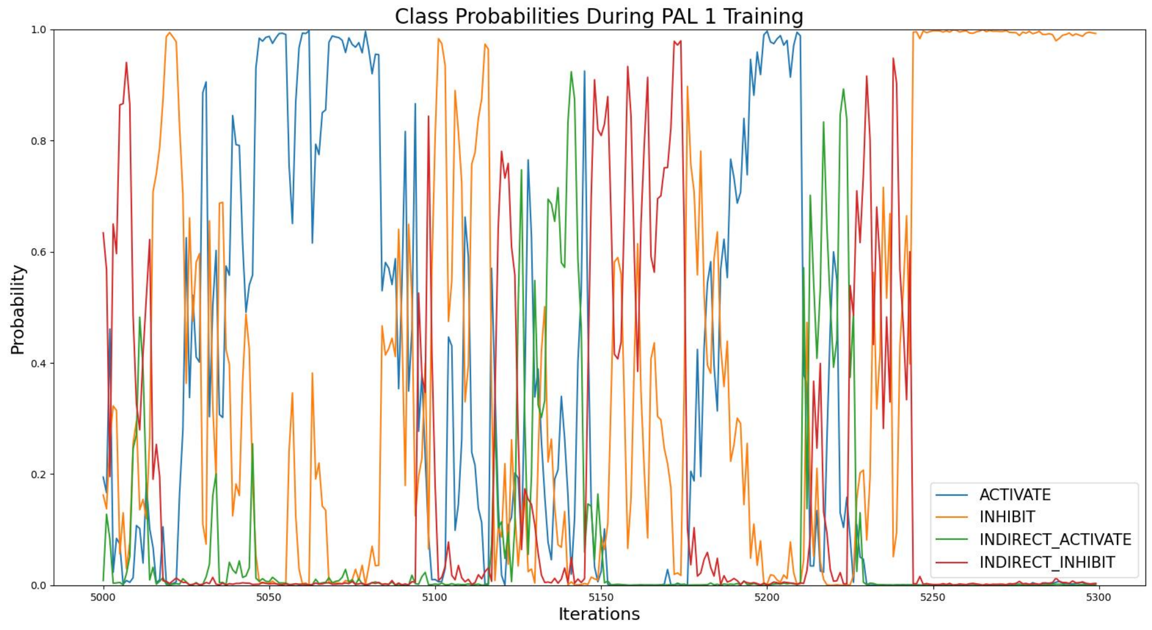


Figure 5.4: Visualization of class probabilities for 300 iterations during PAL 1 training.

From Table 5.1, we also see that PAL 1 had the worst performance by far. When investigating the cause, I observed in Figure 5.4 that the class probabilities behaved similarly to what was hypothesized for this method. There were extended intervals during training where one class dominated the seed probability distribution. This led to periods when other classes were not likely to appear in training at all. For our task, this long-term mode switching during training seemed to hurt generalization.

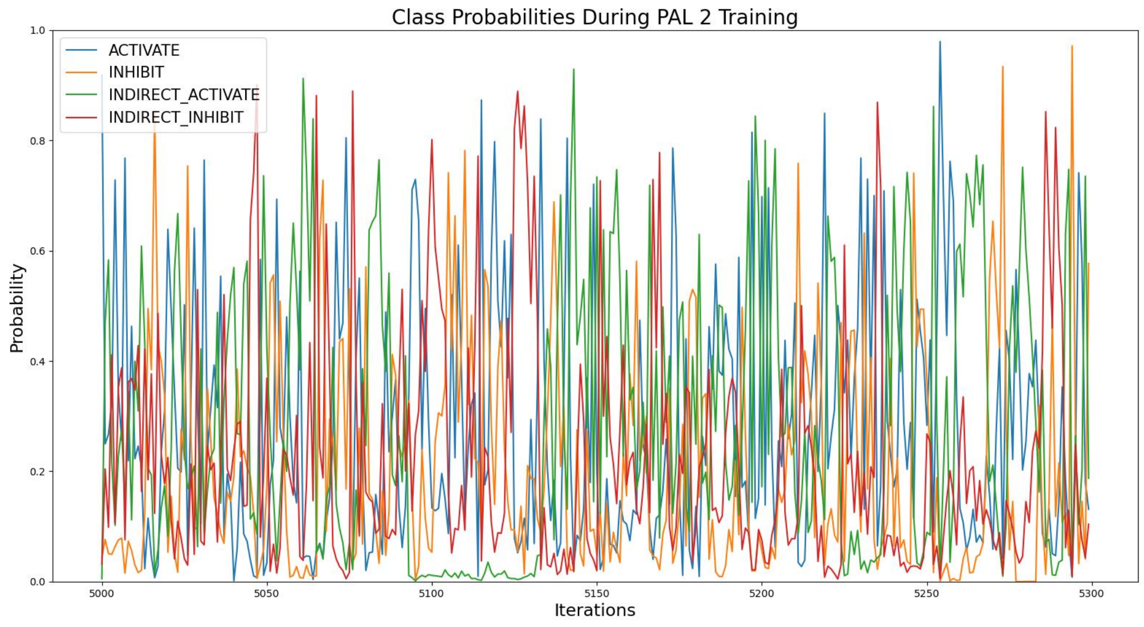


Figure 5.5: Visualization of class probabilities for 300 iterations during PAL 2 training.

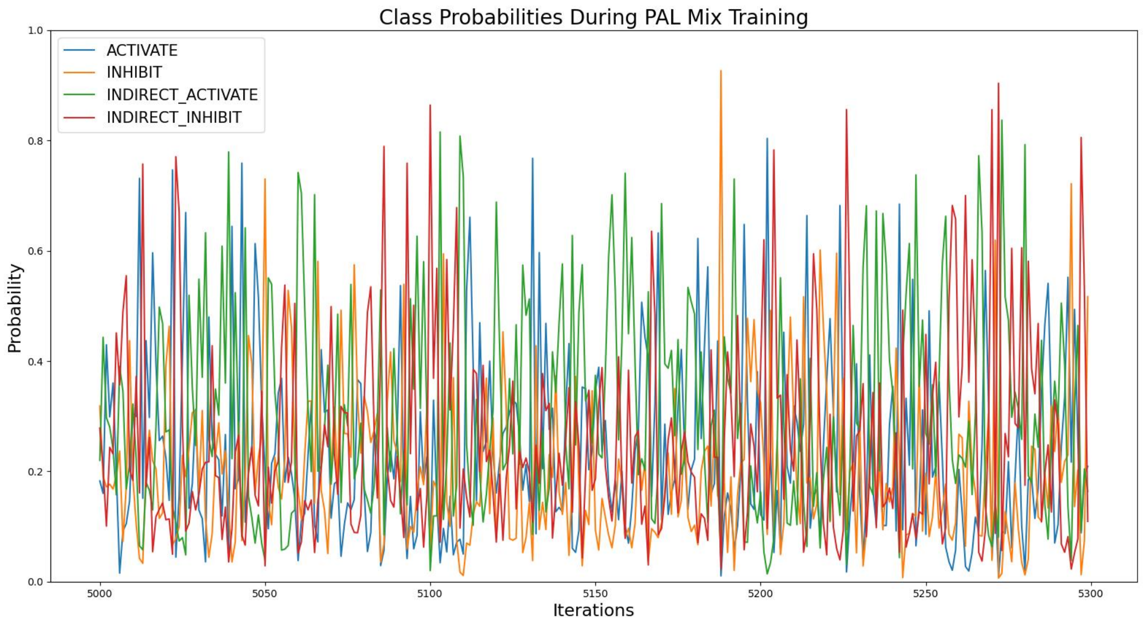


Figure 5.6: Visualization of class probabilities for 300 iterations during PAL Mix training.

Despite the disadvantages of PAL 1, combining its formulation with PAL 2 in PAL Mix seemed to stabilize its performance. This validates our hypothesis that we could combine both methods to leverage the benefits from each while mitigating their disadvantages. Surprisingly, PAL 2’s formulation for putting more emphasis on just the target class did not hurt its performance. This may partially be because we are using a mixed diversity and uncertainty sampling approach. That is to say, even if we put more emphasis on one class over another as in PAL 2, the class distributions were not skewed enough to prevent other classes from appearing. As observed in Figure 5.5, the class distributions for 300 iterations during training are more intertwined. Even if one class was dominating for several iterations, the bottom floor for all other classes was not pushed all the way to zero as in Figure 5.4. This may allow us to learn on specific classes in a more targeted fashion, without completely sacrificing our sampling diversity. In PAL Mix, the floor and ceiling are compressed for the class probabilities (Figure 5.6), which may not allow targeted class specific training to the same extent. However, more experiments would be necessary to fully explore these differences in training dynamics and their benefits.

5.3.2 Extending PAL with Momentum

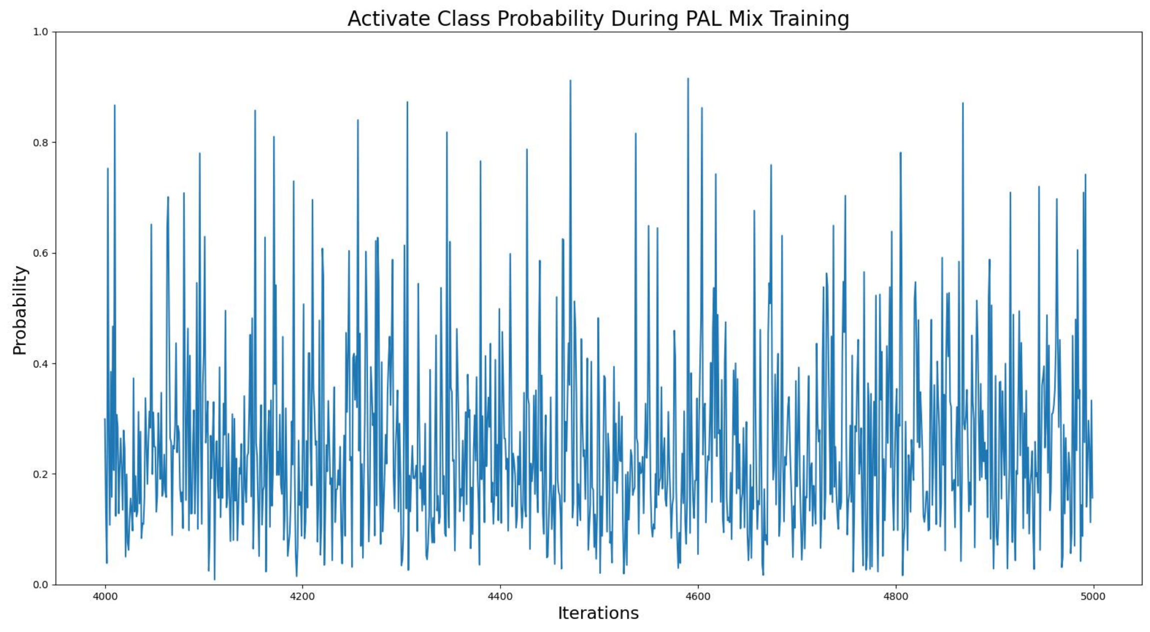


Figure 5.7: Plot of the ‘activation’ class probability for 300 iterations during PAL Mix training.

After validating the PAL training schema, I also looked to further extend our schema. To do that, I investigated if we should take the updated probabilities directly or not. As observed in Figure 5.7, taking the probabilities directly can cause a rapid shift in class probabilities from iteration to iteration. This may not always be ideal, as it may not give the model enough time to learn the features from prior loss and class information. To better understand if this was a beneficial property, I needed to test mixing previous values’ information and new information. Fortunately, this is a well explored topic in optimizers [65] and data fusion [66].

$$\mathbf{V_t} = \beta * \mathbf{V_{t-1}} + (1-\beta) * \mathbf{S_t} \quad (5.1)$$

$$\mathbf{P_t} = \mathbf{P_{t-1}} + \mathbf{V_t} \quad (5.2)$$

To better understand how long a model should learn on a given distribution, I combined the PAL training schema with a simple momentum term (5.1,5.2). Specifically, I kept a momentum vector V_t that maintains previous update information for each class. V_t is initialized to a zero vector. β is a hyperparameter that decides how much to listen to previous updates and is fixed between 0 and 1. Notably, starting with all of the class probabilities to be equal and using a β of 1.0 reduces to our standard training schema and using a β of 0.0 reduces to our basic PAL setting. For my experiments, I trained RetinaNet with PAL Mix in the same fashion as before for 4 classes (activate, inhibit, indirect activate, and indirect inhibit), but with different values for the momentum term. Again, I train 10 models for each momentum value.

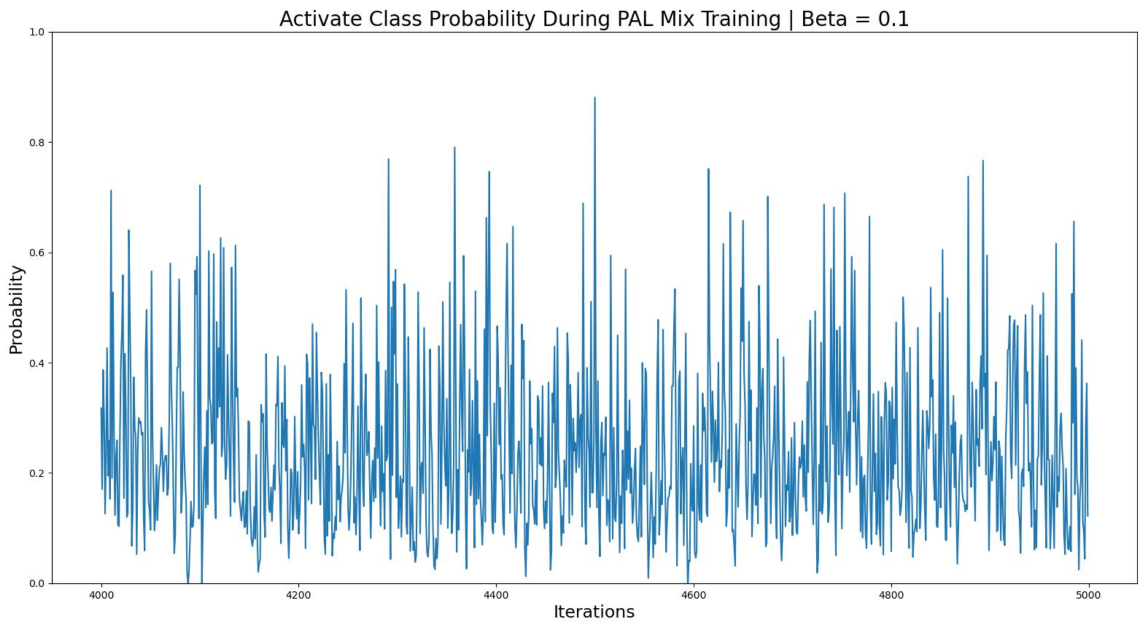


Figure 5.8: Plot of the 'activation' class probability for 300 iterations during PAL Mix training with a momentum factor of 0.1.

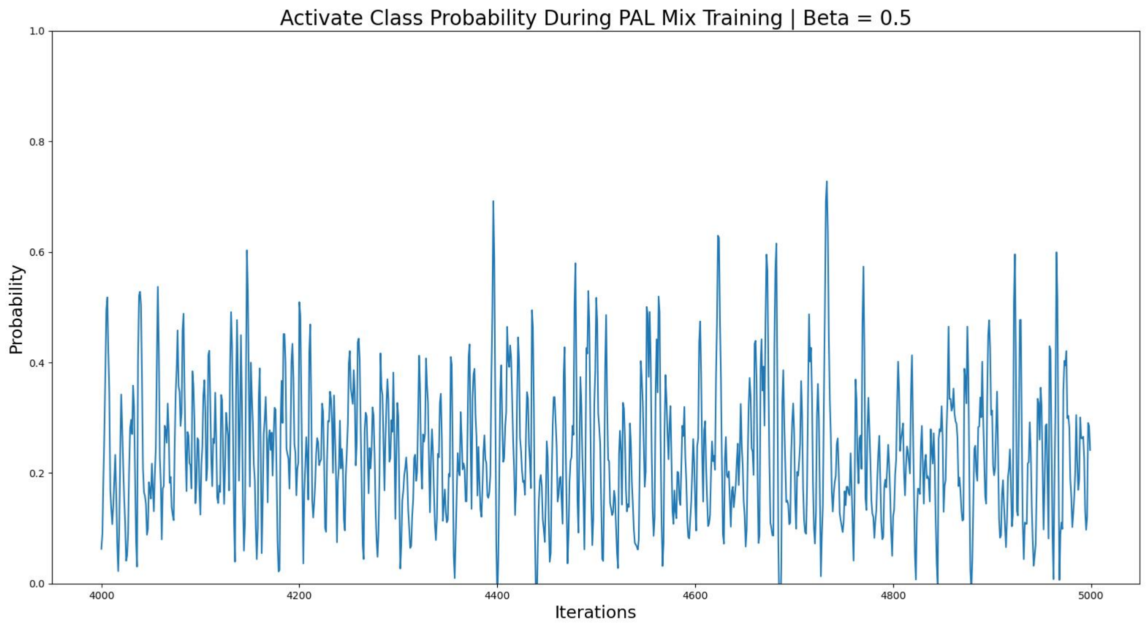


Figure 5.9: Plot of the 'activation' class probability for 300 iterations during PAL Mix training with a momentum factor of 0.5.

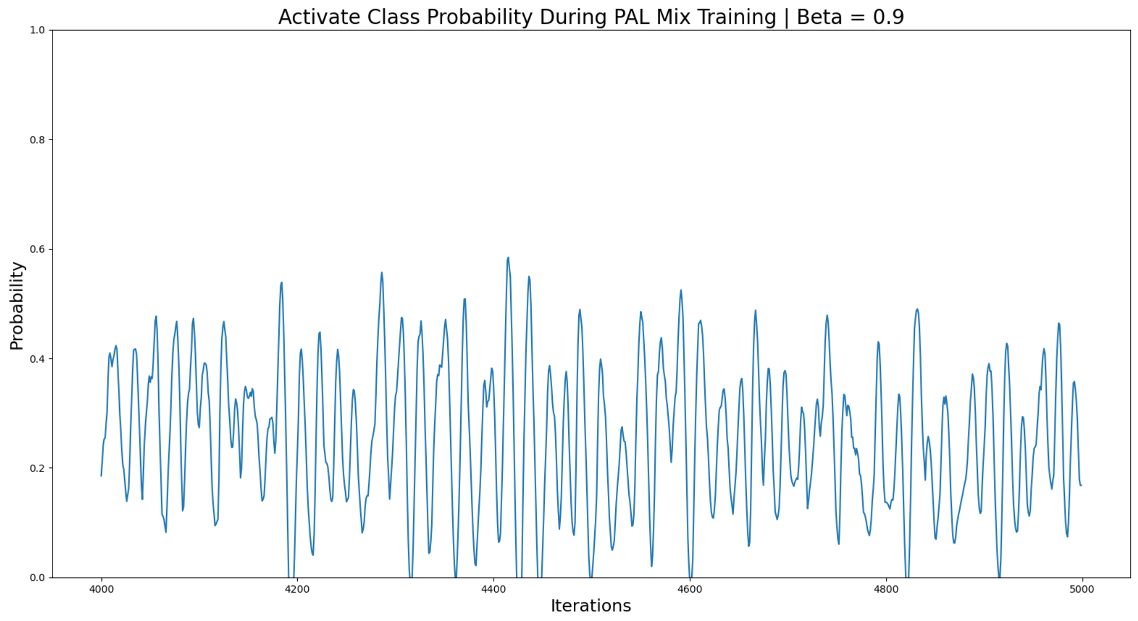


Figure 5.10 Plot of the ‘activation’ class probability for 300 iterations during PAL Mix training with a momentum factor of 0.9.

Table 5.2: Testing performance of PAL trained models with momentum in terms of precision, recall, and average number of iterations to model convergence.

	β	mAP	mAR	Avg. Iterations
Standard Training	1.0	42.7 +/- 1.3	81.7 +/- 5.3	9860 +/- 1564
PAL Mix	0.0	41.8 +/- 1.6	83.5 +/- 3.0	8900 +/- 1790
PAL Mix	0.1	42.1 +/- 1.7	81.2 +/- 1.49	**8460 +/- 1111**
PAL Mix	0.5	**44.0 +/- 2.7**	**83.5 +/- 2.8**	9860 +/- 1383
PAL Mix	0.9	42 +/- 3.7	83.2 +/- 3.3	8620 +/- 2112

As seen in Figures 5.8, 5.9, and 5.10, incorporating a momentum term has the intended effect of smoothing out changes in the probability distribution for a given class. From Table 5.2, one can also see that a β of 0.5 improves performance over baseline in the same number of iterations. Additionally, a momentum of 0.1 obtains almost the same performance as standard training but in even fewer iterations than PAL Mix or PAL 2 and with less variability in iterations. A momentum factor of 0.9 showed decent average performance in recall and precision and number of iterations required for convergence, but also had the most variability. A large momentum factor works well in gradient descent, since the optimal value (global minima) does not move during training. However, in our case, the optimal sample to be generated from iteration to iteration does change constantly. In this way, having a small momentum allows us to leverage previous knowledge to some extent while still updating the seed distributions rapidly.

5.4 Summary

In this chapter, I explained two schools of thought for learning from error when incorporating our synthetic data into active learning. I also demonstrated how to combine these two methods into PAL Mix. Additionally, I illustrated how PAL 2 and PAL Mix were able to balance recall, precision, and runtime compared to traditional training and conventional active learning. I further showed how to extend PAL by incorporating a momentum term and the benefit that such factor has to further speed up training or improve generalization.

Chapter 6

Future Work & Conclusions

In this chapter I will review the key contributions of this thesis and explore several avenues for further research.

6.1 Future Work

6.1.1 Gene-Extraction

As our pipeline involves multi-stage processing, each of these modules can be improved to boost our overall extraction. For our gene and indicator localization, we plan to unify our head and body extraction into a single model. We were unable to do this previously with RetinaNet due to a lack of data. With access to more data, we now also want to experiment with other object detection models like DETR [27] and extend to more relationship types. Additionally, we plan to replace the google OCR module with an open-source OCR system, since we have collaborators in China who do not have access to this google service. We also want to improve upon our relationship pairing strategy, since our current method struggles to correctly identify the correct relationships when clusters are involved and those with very small indicators. To add more confidence in our extracted relationships, we also want to score each of the interactions we extract. We could do this scoring by assigning weights to the conference or publication sources we pull from or by leveraging text information to cross-validate our image triplets with text triplets.

6.1.2 Synthetic Data

We also have several ideas to further improve our synthetic data. While we can mimic the relationships from pathway figures, we also want them to be placed in more realistic settings. Measuring the benefits from incorporating basic structured and unstructured noise indicated some shortcoming in this respect. By introducing more realistic and diverse contexts for our relationships, we could further bridge this gap. Additionally, we want to generalize our generative algorithm so that it can be more easily applied to settings beyond pathway diagrams and documents.

6.1.3 Active Learning

There are several directions for further improvements and validation for our active learning schema as well. We see immediate applicability of our PAL schema for OCR training, which would help further help validate our method as well. We also want to robustly test the upper bound of our PAL training schema by looking at training longer, with different learning rates, and repeating our experiments to ensure the differences between methods are statistically significant. Additionally, while the extended mode switching from PAL 1 did not work well, it would be interesting to explore how short-term mode switching guided by the loss values could impact training. Such an approach could be implemented using PAL with an Adam [67] style update equation rather than a single momentum term. Exploring these training dynamics would be very important for better understanding PAL and guide further improvements.

6.2 Conclusions

The goal of this thesis was to develop new methods for augmenting biomedical literature curation. To that end, I helped develop a gene relationship extraction pipeline that leverages natural language and image processing. While our pipeline was able to extract gene interactions, the lack of generalizing object detection models limited our extraction efforts. To improve our object detection localization and overcome diminishing returns from traditional augmentations on our pathway figures, I developed a rule-based generative algorithm for creating new pathway diagrams for training. This synthetic data generation was able to successfully improve generalization in mixed batch settings with augmented data and showed strong standalone performance when additional noise was incorporated. To further explore how to best leverage our ability to generate synthetic samples, I investigated how to incorporate our synthetic data into an active learning schema. Our Pool-less Active Learning (PAL) training framework leverages the classification loss from a current training batch to generate the next batch of samples for training. This novel approach demonstrated good balance between recall, precision, and runtime compared to traditional training and least confidence active learning.

Bibliography

[1] Lu, Ling, et al. "Role of SMAD and non-SMAD signals in the development of Th17 and regulatory T cells." *The Journal of immunology* 184.8 (2010): 4295-4306.

[2] Memon, Jamshed, et al. "Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR)." IEEE Access 8 (2020): 142642-142668.

[3] Bowen, Glenn A. "Document analysis as a qualitative research method." Qualitative research journal (2009).

[4] D. Shin, G. Arthur, M. Popescu, D. Korkin, and C. R. Shyu, "Uncovering influence links in molecular knowledge networks to streamline personalized medicine," Journal of Biomedical Informatics, vol. 52, no. E95.A, pp. 394-405, 2014.

[5] I. S. f. Biocuration, "Biocuration: Distilling data into knowledge," PLoS biology, vol. 16, no. 4, p. e2002846, 2018.

[6] B. M. Kuenzi and T. Ideker, "A census of pathway maps in cancer systems biology," Nature Reviews Cancer, vol. 20, no. 4, pp. 233-246, 2020.

[7] K. Z. Vardakas, G. Tsopanakis, A. Poulopoulou, and M. E. Falagas, "An analysis of factors contributing to PubMed's

growth," Journal of Informetrics, vol. 9, no. 3, pp. 592-617, 2015.

[8] C. Li, M. Liakata, and D. Rebholzschuhmann, "Biological network extraction from scientific literature: state of the art and challenges," Briefings in Bioinformatics, vol. 15, no. 5, pp. 856-877, 2014.

[9] P. D. Karp, "Can we replace curation with information extraction software?," Database, vol. 2016, 2016.

[10] J. Szostak et al., "Construction of biological networks from unstructured information based on a semi-automated curation workflow," Database the Journal of Biological Databases & Curation, vol. 2015, p. bav057, 2015.

[11] S. Ananiadou, P. Thompson, R. Nawaz, J. Mcnaught, and D. B. Kell, "Event-based text mining for biology and functional genomics," Briefings in Functional Genomics, vol. 14, no. 3, pp. 213-230, 2015.

[12] J. Li et al., "BioCreative V CDR task corpus: a resource for chemical disease relation extraction," Database the Journal of Biological Databases & Curation, vol. 2016, p. baw068, 2016.

[13] Z. Ahmed, S. Zeeshan, and T. Dandekar, "Mining biomedical images towards valuable information retrieval in biomedical and life sciences," Database, vol. 2016, 2016.

[14] D. Kim and H. Yu, "Figure text extraction in biomedical literature," PloS one, vol. 6, no. 1, p. e15338, 2011.

[15] S. Xu and M. Krauthammer, "Boosting text extraction from biomedical images using text region detection," in Biomedical Sciences & Engineering Conference, 2011.

[16] A. Riutta, K. Hanspers, and A. R. Pico, "Identifying Genes in Published Pathway Figure Images," bioRxiv, p. 379446, 2018.

[17] S. Kozhenkov and M. Baitaluk, "Mining and integration of pathway diagrams from imaging data," Bioinformatics, vol. 28, no. 5, pp. 739-742, 2012.

[18] D. W. Fei He, Yulia Innokenteva, Olha Kholod, Dmitriy Shin and Dong Xu, "Extracting Molecular Entities and Their Interactions from Pathway Figures Based on Deep Learning," Proceedings of ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB’19). Niagara Falls, NY, USA, 2019, doi: https://doi.org/10.1145/3307339.3342187.

[19] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[20] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

[21] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems 28 (2015).

[22] Lin, Tsung-Yi, et al. "Feature pyramid networks for object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[23] Lin, Tsung-Yi, et al. "Focal loss for dense object detection." Proceedings of the IEEE international conference on computer vision. 2017.

[24] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[25] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[26] Du, Xianzhi, et al. "Spinenet: Learning scale-permuted backbone for recognition and localization." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.

[27] Carion, Nicolas, et al. "End-to-end object detection with transformers." European conference on computer vision. Springer, Cham, 2020.

[28] Bjerrum, Esben Jannik. "SMILES enumeration as data augmentation for neural network modeling of molecules." arXiv preprint arXiv:1703.07076 (2017).

[29] Duong, Huu-Thanh, and Vinh Truong Hoang. "Data Augmentation Based on Color Features for Limited Training Texture Classification." 2019 4th International Conference on Information Technology (InCIT). IEEE, 2019.

[30] DeVries, Terrance, and Graham W. Taylor. "Improved regularization of convolutional neural networks with cutout." arXiv preprint arXiv:1708.04552 (2017).

[31] Zhang, Hongyi, et al. "mixup: Beyond empirical risk minimization." arXiv preprint arXiv:1710.09412 (2017).

[32] Yun, Sangdoo, et al. "Cutmix: Regularization strategy to train strong classifiers with localizable features." Proceedings of the IEEE/CVF international conference on computer vision. 2019.

[33] Shanthamallu, Uday Shankar, et al. "A brief survey of machine learning methods and their sensor and IoT applications." 2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA). IEEE, 2017.

[34] Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).

[35] Wang, Ting-Chun, et al. "High-resolution image synthesis and semantic manipulation with conditional gans." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

[36] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." arXiv preprint arXiv:1312.6114 (2013).

[37] Jaderberg, Max, Karen Simonyan, and Andrew Zisserman. "Spatial transformer networks." Advances in neural information processing systems 28 (2015).

[38] Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." arXiv preprint arXiv:1508.06576 (2015).

[39] F. Wang, Y. Zhuang, H. Gu and H. Hu, "Automatic Generation of Synthetic LiDAR Point Clouds for 3-D Data Analysis," in IEEE Transactions on Instrumentation and Measurement, vol. 68, no. 7, pp. 2671-2673, July 2019, doi: 10.1109/TIM.2019.2906416.

[40] Griffiths, David, and Jan Boehm. "SynthCity: A large scale synthetic point cloud." arXiv preprint arXiv:1907.04758 (2019).

[41] Lewis, David D., and William A. Gale. "A sequential algorithm for training text classifiers." SIGIR’94. Springer, London, 1994.

[42] Lewis, David D., and Jason Catlett. "Heterogeneous uncertainty sampling for supervised learning." Machine learning proceedings 1994. Morgan Kaufmann, 1994. 148-156.

[43] Scheffer, Tobias, Christian Decomain, and Stefan Wrobel. "Active hidden markov models for information extraction." International Symposium on Intelligent Data Analysis. Springer, Berlin, Heidelberg, 2001.

[44] Shannon, Claude Elwood. "A mathematical theory of communication." The Bell system technical journal 27.3 (1948): 379-423.

[45] Freund, Yoav, et al. "Selective sampling using the query by committee algorithm." Machine learning 28.2 (1997): 133-168.

[46] Pop, Remus, and Patric Fulop. "Deep ensemble bayesian active learning: Addressing the mode collapse issue in monte carlo dropout via ensembles." arXiv preprint arXiv:1811.03897 (2018).

[47] Houlsby, Neil, et al. "Bayesian active learning for classification and preference learning." arXiv preprint arXiv:1112.5745 (2011).

[48] Yoo, Donggeun, and In So Kweon. "Learning loss for active learning." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.

[49] Kirsch, Andreas, Joost Van Amersfoort, and Yarin Gal. "Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning." Advances in neural information processing systems 32 (2019).

[50] Zhdanov, Fedor. "Diverse mini-batch active learning." arXiv preprint arXiv:1901.05954 (2019).

[51] Ash, Jordan T., et al. "Deep batch active learning by diverse, uncertain gradient lower bounds." arXiv preprint arXiv:1906.03671 (2019).

[52] Settles, Burr. "Active learning literature survey." (2009).

[53] Ren, Pengzhen, et al. "A survey of deep active learning." ACM Computing Surveys (CSUR) 54.9 (2021): 1-40.

[54] Nielsen, Christopher, and Michal M. Okoniewski. "GAN Data Augmentation Through Active Learning Inspired Sample Acquisition." CVPR Workshops. 2019.

[55] Hong, SeulGi, et al. "Deep Active Learning with Augmentation-based Consistency Estimation." arXiv preprint arXiv:2011.02666 (2020).

[56] Tran, Toan, et al. "Bayesian generative active deep learning." International Conference on Machine Learning. PMLR, 2019.

[57] Kim, Yoon-Yeong, et al. "LADA: Look-Ahead Data Acquisition via Augmentation for Deep Active Learning." Advances in Neural Information Processing Systems 34 (2021).

[58] Agarap, Abien Fred. "Deep learning using rectified linear units (relu)." arXiv preprint arXiv:1803.08375 (2018).

[59] C. H. Wei, A. Alexis, L. Robert, and Z. Lu, "PubTator central: automated concept annotation for biomedical full text articles," Nuclc Acids Research, no. W1, p. W1, 2019.

[60] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," Dokl. Akad. Nauk SSSR, 1965, 1966.

[61] J. Shi, "Good Features to Track," in Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 1994, 1994.

[62] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "LabelMe: a database and web-based tool for image annotation," International journal of computer vision, vol. 77, no. 1-3, pp. 157-173, 2008.

[63] Cooley, James W., and John W. Tukey. "An algorithm for the machine calculation of complex Fourier series." Mathematics of computation 19.90 (1965): 297-301.

[64] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009.

[65] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016).

[66] Bleiholder, Jens, and Felix Naumann. "Data fusion." ACM computing surveys (CSUR) 41.1 (2009): 1-41.

[67] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).