Parsing Genetic Models

by

Michael Natale Lombardo

A thesis submitted to the

School of Graduate and Postdoctoral Studies

in partial fulfillment of the requirements for the degree of

Masters of Science in **Computer Science**

Faculty of Science

University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

January, 2022

Thesis Examination Information

Submitted by: **Michael Natale Lombardo**

Master of Science in **Computer Science**

Thesis title: Parsing Genetic Models

An oral defense of this thesis took place on December, 8, 2021 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Patrick Hung
Research Supervisor	Dr. Faisal Z. Qureshi
Examining Committee Member	Dr. Nicholas J. Provart
Thesis Examiner	Dr. Bill Kapralos

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Applications of computer vision have seen great success recently, yet there are few approaches dealing with visual illustrations. We propose a collection of computer vision applications for parsing genetic models. Genetic models are a visual illustration often used in the biological sciences literature. These are used to demonstrate how a discovery fits into what is already known about a biological system. A system that determines the interactions present in a genetic model can be valuable to researchers studying such interactions. The proposed system contains three parts. First, a triplet network is deployed to decide whether or not a figure is a genetic model. Second, a popular object detection network YOLOv5 is trained to locate regions of interest within genetic models using various deep learning training techniques. Lastly, we propose an algorithm that can infer the relationships between the pairs of genes or textual features present in the genetic model.

Keywords: Diagram Understanding; Diagram Detection; Visual Illustrations; Bioinformatics; Object Detection.

Author's Declaration

______________________________ Michael Natale Lombardo

Statement of Contributions

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication. I have used standard referencing practices to acknowledge ideas, research techniques, or other materials that belong to others. Furthermore, I hereby certify that I am the sole source of the creative works and/or inventive knowledge described in this thesis.

Acknowledgements

A sincere thank you to my family and friends who I hold closest, your motivation kept me driven to always be hungry to succeed. The great times I have cherished with all of you constantly reminds me of the immense support offered to my success in academics.

I would like to thank Ontario Tech University and Faculty of Science for funding my Masters studies, supplemented by the Ontario Graduate Scholarship award. Additionally, a generous contribution from Genome Canada to support funding for this work.

Thank you to the members of the University of Toronto's Provart Lab. Their advice and contributions on biology and bioinformatics topics made this work possible.

A special acknowledgement goes to Dr. Faisal Z. Qureshi, for supervising my Masters thesis. His advice, mentorship, and ability to always see the positive in a scramble of work always helped steer me in the right direction throughout my studies. Although our journey together was never easy, you always were willing to let me take chances with my ambitious ideas.

"You are what you choose to be." – The Iron Giant 1999

"The greatest adventure is what lies ahead." – J.R.R Tolkien

Table of Contents

List of Figures

List of Tables

List of Abbreviations

NCBI	National Center for Biotechnology Information
LSTM	Long Short-Term Memory
BAR	Bio-Analytic Resource for Plant Biology
YOLO	You Only Look Once
OCR	Optical Character Recognition
API	Application Programming Interface
GN	GeneNet
GNS	GeneNetSyn
V	Variable
DP	Diagram Parsing
GN-DP	GeneNet Diagram Parsing Dataset
DA	Domain Adapatation
FT	Fine-Tune
TL	Transfer Learning
AI2	Allen Institute for Artificial Intelligence
AI2D	Allen Institute for Artificial Intelligence Diagrams
DPG	Diagram Parse Graph
DSDP-Net	Deep Sequential Diagram Parser
DQA-Net	Diagram Question Answering System
DGGN	Dynamic Graph Generation Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
MNIST	Modified National Institute of Stands and Technology
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
COCO	Common Objects in Context
SSD	Single Shot Multi Box Detector
R-CNN	Region Based Convolutional Neural Networks
VGG	Very Deep Convolutional Networks
ACM	Active Contour Model
LR	Left-to-Right
RL	Right-to-Left
RGB	Red Green Blue

Chapter 1 Introduction

This thesis develops a computer vision system for constructing textual descriptions from genetic models shown in Figure 1.1. Genetic models are a type of visual illustration often used within biological sciences to capture the relationship between two or more genes or proteins. Genetic models are widely used in publications by biological researchers to convey their contribution regarding a particular biological system. Similar genetic models are commonly found as figures in biology publications. The publication figures used throughout this thesis are found on National Center for Biotechnology Information PubMed using the search query: *Arabidopsis thaliana*.

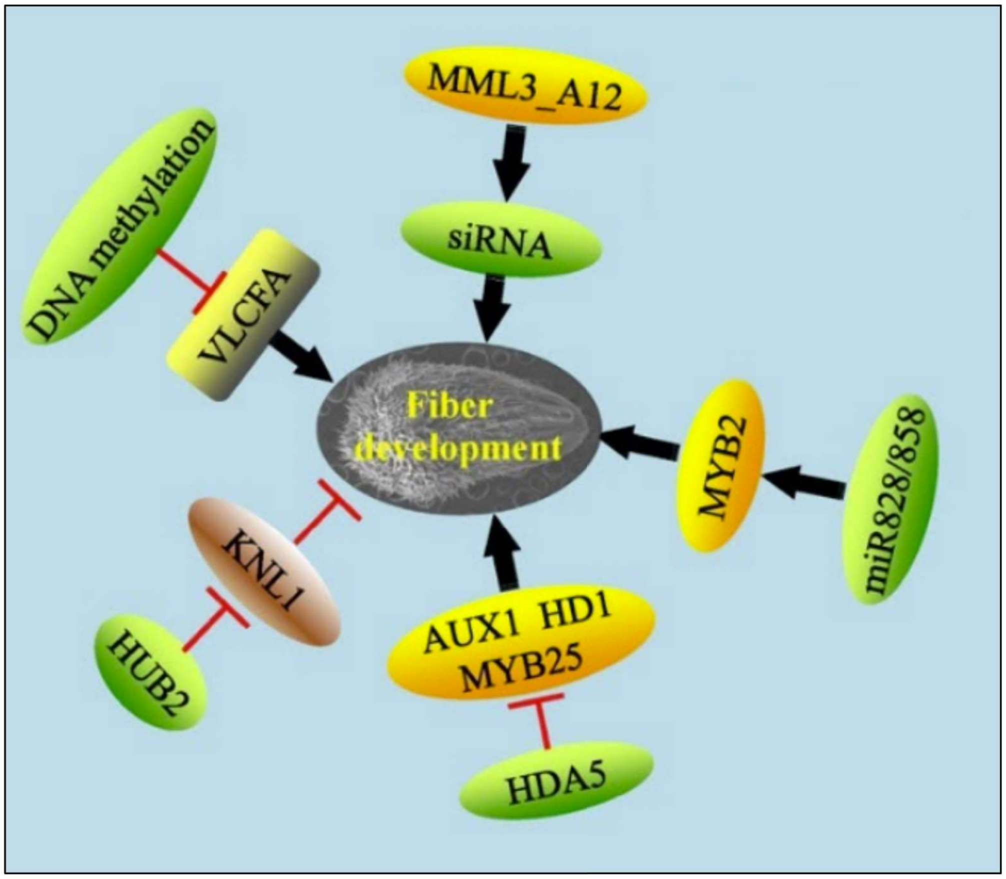


Figure 1.1. Publication figure extracted from National Center for Biotechnology Information PubMed with the search query *Arabidopsis thaliana*. Genetic model from Wang *et al.* [1].

Computer vision is an active area of research and a large body of work focuses on image and video analysis and understanding natural imagery. Still, there is little prior work on computer vision systems for analyzing visual illustrations and more specifically, for understanding genetic models. We consider the previous work of diagram understanding to be the most comparable to our work, yet consider our work diagram parsing as we do not conduct question-answering. Work published in the 1990s [23, 24, 25, 26] are notable exceptions. In 2016, Kembhavi *et al.* proposed a computer vision system that focuses on question answering for grade-school science diagrams. Their approach proposed a Long Short-Term Memory (LSTM)-based network for creating diagram parse graphs, which relied heavily on contextual information that is not made publicly available. Despite our best efforts, we found that these methods [2] are not reproducible using the dataset and tools available to us. These methods do not seem to be well-suited for our problem of constructing textual descriptions from genetic models. Two key reasons why previous works in diagram understanding cannot be applied to our task are: (1) These focus on question-answering and (2) these models assume contextual information available in the images.

Within this context, this thesis develops a computer vision system for (1) detecting genetic models and (2) constructing textual descriptions from these genetic models. Step (1) is important since biological publications often contain multiple images, most of which are not genetic models. Only genetic models are subsequently processed in step (2).

1.1 Motivation

The motivation of this work comes from discussions at a recent Multinational Arabidopsis Steering Committee [27] that suggest a desire within the plant biology community to have access to better tools for data mining information from existing papers. For example, one user commented, “Can we make the literature database easier to automatically mine for the content of papers?” Another user wrote, “I think we need more text and data mining sites." Someone chimed, "We'd like to easily search new literature and datasets for connections." While it is possible to utilize *data mining text analysis* techniques to search within these publications, currently we lack necessary tools to analyze and make sense of graphical elements occurring within these papers. Genetic models are but one such element. Some other examples of the graphical elements found in publications include graphs and images. This work aims to use computer vision technology capable of parsing genetic models to create more powerful tools for biological researchers than are currently available. The classification network from our proposed computer vision system to reason about genetic models developed in this thesis is already being used to support the search functionality available within University of Toronto's Bio-Analytic Resource for Plant Biology (BAR).

Current popular bioinformatics works focus on the visualization such as ePlant [28]; a suite of open-source worldwide web-based tools for visualization of datasets from the organism *Arabidopsis thaliana.* Given the aforementioned computer vision system which extracts textual descriptions from genetic models, bioinformaticians can use this information for visualization purposes. Visualizations can include showing relationships between a wide collection of genes, while also referring directly to publications and genetic models where this information is found.

The work introduced in this thesis can extract information from genetic models, which can be used on plant biological systems and gene search engines to recommend additional genes or proteins which have inherent relationships within genetic models. We propose a system that, for example, given a search query of "Gene A", we will be capable of providing any relationships associated with "Gene A". This would produce a resulting output for this search query: "Gene A activates Gene B"; "Gene A inhibits Gene C"; and "Gene D activates Gene A". The system can be considered low-risk as its purpose is to assist researchers in discovering other associated genes from their original search. Any biological researcher using a search engine of this nature would naturally have their own validation to ensure our recommendations of associated genes would be correct through

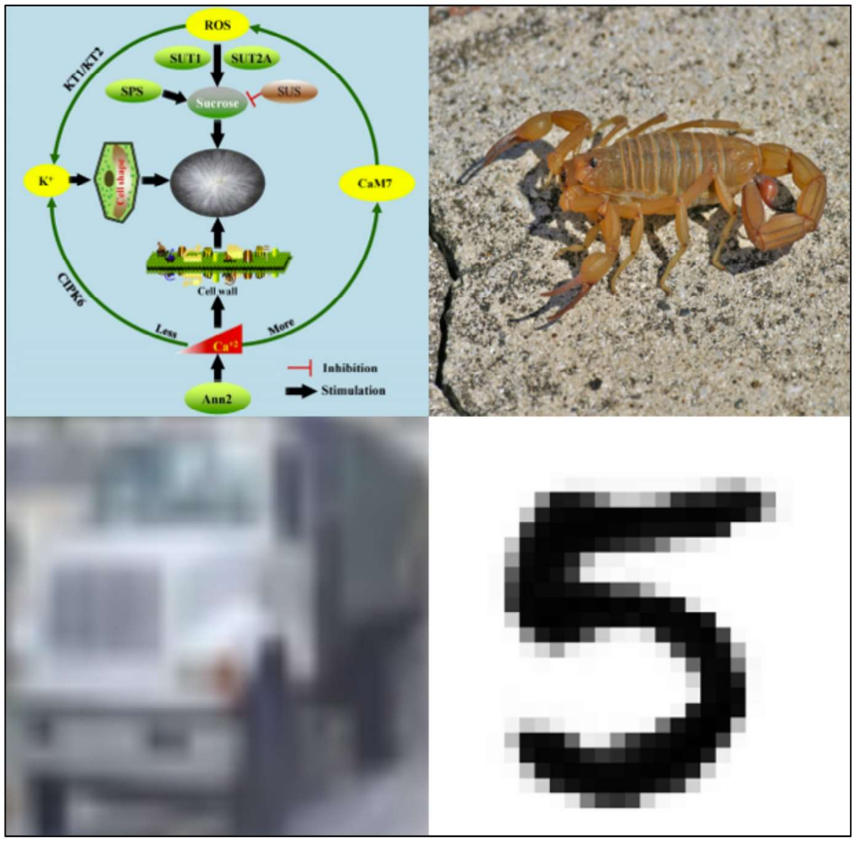


Figure 1.2. Genetic models (top-left) exhibit different visual characteristics than images typically used for training computer vision deep networks. Clock-wise from top-right: images from CIFAR-10, ImageNet, and MNIST datasets. Genetic model from Wang *et al.* [1].

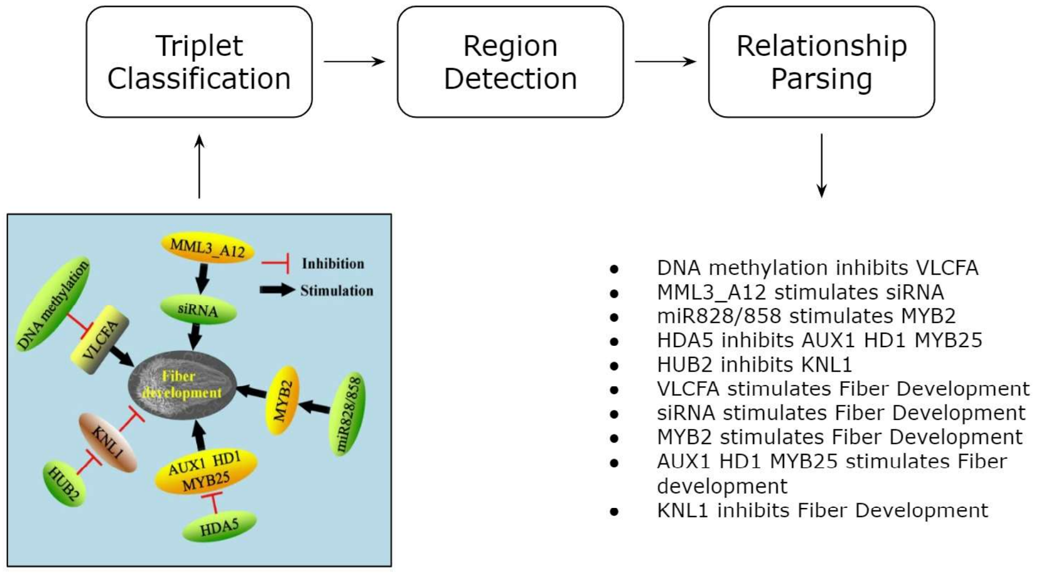
visualizations of the resulting relationships.

1.2 Datasets

We have developed deep learning-based methods for genetic model detection and analysis. Deep learning methods are data-dependent. Specifically, developing deep learning methods for extracting relationships among genes found within a genetic model requires access to richly annotated genetic models. These annotations must account for any entities found in the genetic model, including genes (textual features) and the relationship between these genes. We found that deep learning models trained on widely available natural image datasets discussed in Section 2.2.1, such as ImageNet, perform poorly on genetic models. This lack of functionality is to be expected.

Genetic models have little in common with natural imagery typically used for training deep learning-based computer vision systems demonstrated in Figure 1.2. These various datasets which are shown in the above figure often contain one verb or region of interest, unlike genetic models which can have many. Therefore, we needed to create genetic model dataset(s) that can be used to develop deep learning detection and analysis models. To this end, the thesis constructs two datasets: (a) a set of genetic models collected from plant biology publications found on PubMed and (b) a scheme for generating richly annotated synthetic datasets that exhibit the visual and semantic cues found in genetic models in (a).

1.3 Overview



- DNA methylation inhibits VLCFA
- MML3_A12 stimulates siRNA
- miR828/858 stimulates MYB2
- HDA5 inhibits AUX1 HD1 MYB25
- HUB2 inhibits KNL1
- VLCFA stimulates Fiber Development
- siRNA stimulates Fiber Development
- MYB2 stimulates Fiber Development
- AUX1 HD1 MYB25 stimulates Fiber development
- KNL1 inhibits Fiber Development

Figure 1.3. Proposed solution for diagram parsing on a genetic model displaying desired output. Broken into three pillars: triplet classification, region detection, and parsing relationships. The goal of this system is to create a textual description of relationships in a genetic model. Genetic model from Wang *et al.* [1]

Figure 1.3 shows the proposed method for constructing textual descriptions from a genetic model. It consists of three steps. First, a Triplet classification network is trained on our synthetic dataset named GeneNetSyn, which classifies whether or not an image is a genetic model. Next, a YOLOv5 model identifies the constituents blocks within genetic models. These blocks consist of genes (geometric shapes enclosing textual elements of gene or protein names) and activation and inhibition lines between these nodes. It is relatively easy to manually create a set of labeled datasets needed to train the classification network in the first step. This dataset simply requires images of genetic models and other publication figures paired with a binary label indicating whether or not an image is a genetic model. Our region detection YOLOv5 model in the second step, however, requires richly annotated datasets, which are burdensome to acquire. Instead, the YOLOv5 model, which was initially trained on natural image datasets, is fine-tuned using synthetic datasets. Furthermore, we employ learning techniques, such as domain adaptation [5, 6, 29] and transfer learning [30]. Before the third step, the system leverages Google OCR API discussed in Appendix A.3 to extract textual elements from the genetic models. These extracted textual elements, often being genes and proteins, are presented in a convenient scheme to index the genetic model within the University of Toronto's BAR. Lastly, the thesis develops an activation/inhibition line analysis system to extract a textual description of the form, "Gene A inhibits Gene B."

1.4 Contributions

A system that can reason about genetic models is desirable. Our work described throughout this thesis introduces a system capable of improving the quality and efficiency of data mining and literature review in the field of biology. This can be made possible by implementing the proposed system into a search engine and using the results for each image to recommend additional genes of interest. For example, if a researcher is to launch a search query for gene A, and a relationship is present between gene A and gene B, the system would recommend the researcher to also look at gene B. We leverage applications of computer vision to increase the capability of search engines within publication libraries. The contributions made in this thesis include the following:

- GeneNet-98/500 - Real biological diagrams datasets. A collection of publication figures retrieved from NCBI PubMed on the biological organism *Arabidopsis thaliana*. These are manually annotated with binary labels to distinguish whether or not a diagram is a genetic model.

- GeneNet-DP - An extension of our real biological diagram dataset GeneNet-500. These images were manually annotated for object detection and diagram parsing ground truth to be used as the validation set for the proposed system.
- GeneNetSyn - synthetic biological diagram datasets. These datasets attempt to replicate the visual characteristics of real biological diagrams. These richly annotated datasets contain ground truth of regions of interest using both normalized and darknet standards, textual feature information, and entity-relationship triplets.
- Triplet classification network to identify if a diagram from scholarly works is considered a genetic model.
- Application of object detection from a popular architecture, YOLOv5, trained to identify classes within genetic models using domain adaptation and transfer learning.
- An algorithm for diagram parsing given regions of interest detected from YOLOv5, powered by an energy-minimizing spline guided through external constraint forces approach.

Chapter 2 Related Works

In this chapter, the prior work in diagram understanding and the technical preliminaries required for this thesis will be discussed. First, we outline a collection of the classical diagram understanding techniques and one modern approach on grade-school scientific diagrams. Secondly, we outline the works within computer vision applied on natural imagery used to leverage our parsing of genetic models.

2.1 Prior Works

Diagram understanding is a task which attempts to transcribe a visual illustration for question/answering or research discovery. A visual illustration can be generalized as an image that is used to convey some concept capable of being described textually. A classic example of a visual illustration is the water cycle shown in Figure 2.1. The water cycle can be broken into four fundamental steps: collection, evaporation, condensation, and precipitation. These steps are often depicted with arrows indicating the direction of association such that *collection moves to evaporation*. Although capable of being visually described in many ways, the concept fundamentally will always follow the same steps.

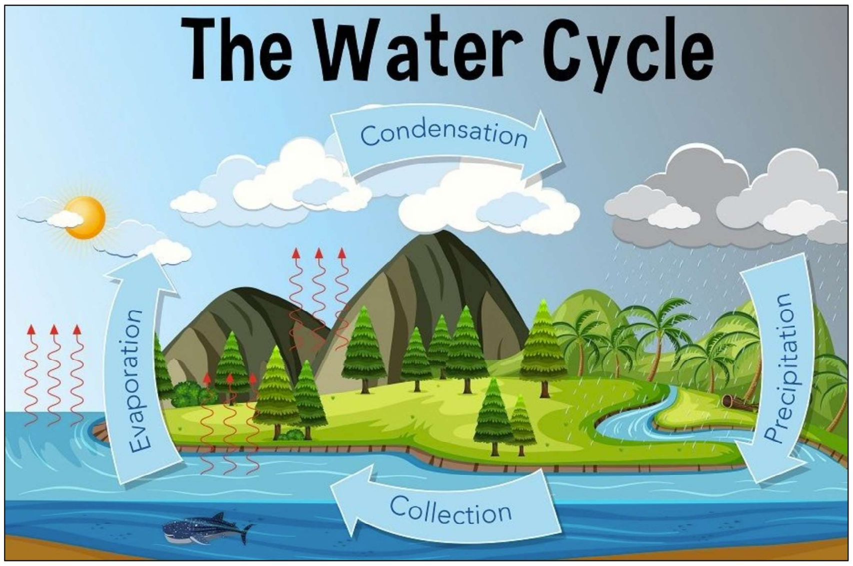


Figure 2.1. Visual illustration demonstrating the water cycle. Courtesy: *https://biologydictionary.net/water-cycle/*

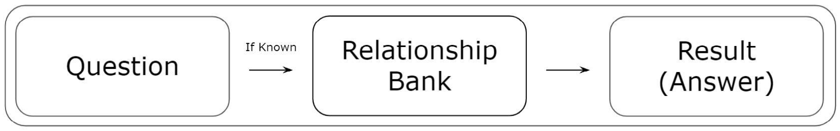


Figure 2.2. Summarized demonstration of classical diagram understanding which uses prior knowledge or data banks to find a solution to a given problem.

Previous approaches of diagram understanding attempt to learn patterns present within concepts to be stored within a data bank shown in Figure 2.2. The task of understanding diagrams was explored in the 1990s [23, 24, 25, 26] using handwritten rules and manual annotations. These systems relied on previous knowledge often stored in data banks or user-given information to assist in finding a solution.

In 2016, Kembhavi *et al.* [2] claimed that visual illustrations fundamentally offer a unique set of challenges to create high-quality representations compared to natural images. Their approach proposes a multiple-choice question answering system for grade-school science diagrams. They define a diagram to be a composite image that consists of graphic space, a set of constituents, and a set of relationships involving these constituents. Visual illustrations offer opportunities for deeper reasoning using applications of computer vision compared to approaches on natural images.

To further the study of diagram understanding, the Paul Allen Institute for Artificial Intelligence (AI2) released a publicly available dataset [2] that consists of 5,000 grade-school science diagrams with 150,000 rich annotations across them. The dataset is used to leverage the tasks of both capturing, and reasoning with information found in diagrams. It also proposes a machine learning model to tackle these tasks. The problem of diagram interpretation and reasoning has two important stages. The first stage is syntactic parsing, which includes detecting and recognizing constituents and their syntactic relationship in a diagram. The second stage includes syntactic interpretation that maps constituents and their relationships to semantic entities and events (real-world concepts).

The concept of diagram parse graphs (DPGs) was first introduced by Kembhavi

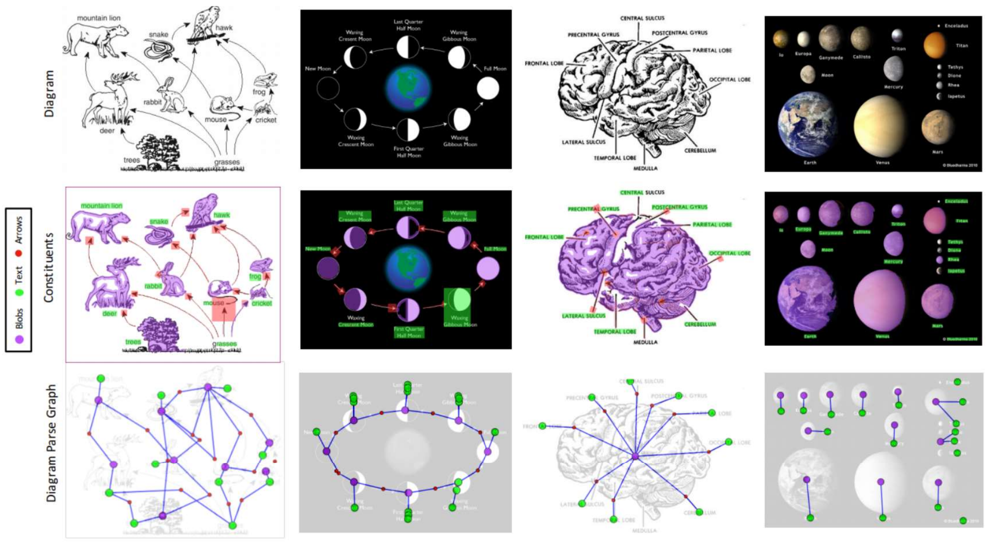


Figure 2.3. Examples of computed diagram parse graphs. Demonstrates the throughput of DSDP-Net on grade-school science diagrams. Courtesy: Kembhavi *et al.* [2]

Constituent	Description
Blobs	Drawings / Illustrative elements
Text Boxes	Titles / Annotations
Arrows	Arrow bodies
Arrow Heads	Arrow heads

Table 2.1. Table showing the various constituent types [2].

el al. [2]. DPGs shown in Figure 2.3 is a representation of a diagram that captures the textual and pictorial constituents of a diagram. Furthermore, their relationships are represented by arrows between constituents. Syntactic diagram parsing is the task of building a diagram parse graph given an image of a diagram. In the context of DPGs, nodes of a graph represent constituents of a diagram, while edges capture relationships between constituents. The authors go on to define four types of constituents shown in Table 2.1 that make up the foundation of a diagram.

Additionally, they go on to propose the Deep Sequential Diagram Parser (DSDP-Net) [2] shown in Figure 2.4, a long short-term memory (LSTM) based model for inferring diagram parse graphs from diagram images. DSDP-Net is a two-stage model that consists of first generating constituent and relationship proposals, and second, presenting the proposals to the DSDP-Net model which builds the DPG iteratively. The first

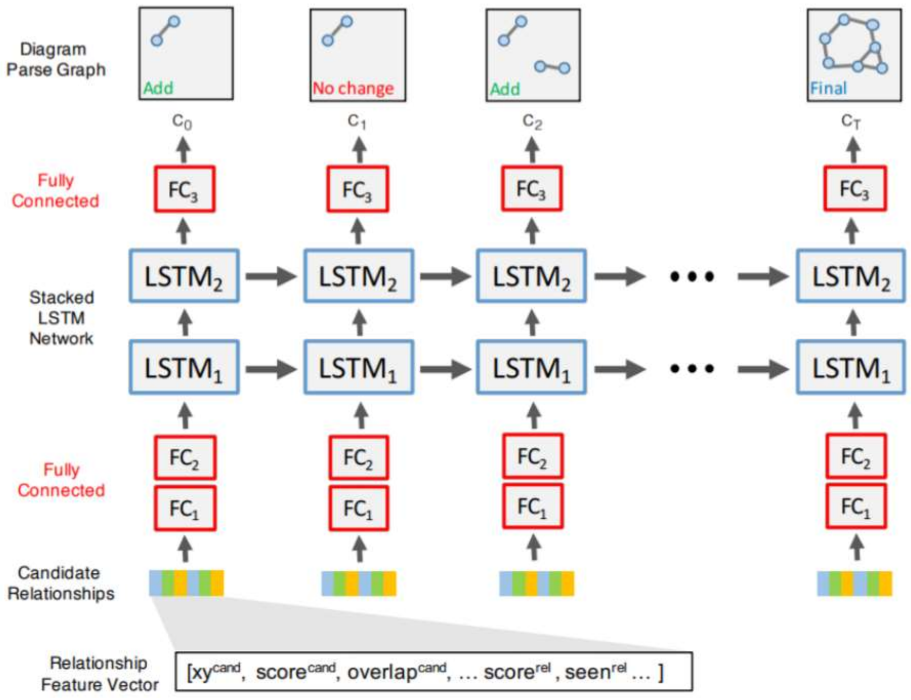


Figure 2.4. Overview of DSDP-Net, an LSTM-based network for inferring diagram parse graph from a given diagram. Courtesy: Kembhavi *et al.* [2]

stage uses a variety of feature extraction methods combined with random forest classifiers for generating the constituent and relationship proposals. The second stage is a 2-layer LSTM that has a hidden state vector consisting of 512 units. One important feature of their approach is the 92-dimensional relationship feature vector which is input for the DSDP-Net. This feature vector contains information including positions, detection score, candidate overlap scores, relationship scores, etc. The important portion of their contribution missing from their publication is the description of the values that the relationship feature vector contains and how they are computed. The lack of description of the relationship feature vector makes this work non-reproducible.

The work done by Kim *el al.* [31] builds on that of Kembhavi *el al.* [2]. Similar to the DSDP-Net, the Dynamic Graph Generation Network (DGGN) is a two-stage model. However, unlike the prior work, both stages can be trained simultaneously in an end-to-end manner. To facilitate this, the first stage of the model is replaced with a single-shot object detection model (a retrained implementation of Single Shot Detector (SSD) [32]) for constituent and proposal generation. These proposals are then sent to the DGGN to parse the DPG. The authors model the DPG using an adjacency matrix, and a recurrent neural network to extract it from the proposals.

These approaches are both are trained and evaluated on the AI2D which contains a large number of richly annotated grade-school science diagrams. DQA-Net proposed by Kembhavi *el al.* [2] offers an effective approach to answering multiple-choice questions given a diagram. One weakness of DQA-Net is the preliminary step of DSDP-Net, which does not identify how their initial data items are processed. This lack of description raises the question of whether there is pre-processing information of the given diagram. Although there is no indication that prior information is used, the question or an information bank sourced by the question given may be used to create values present in the relationship feature vector based on the topic present in the image. DGGN proposed by Kim *et al.* [31] offers a different approach for developing dynamic parse graphs considered diagram graph generation in their work. Their use of a retrained object detection network proves to be an effective approach for identifying constituents. Our approach described throughout this thesis takes inspiration from DGGN, although it does not require previous information to parse the diagram.

Although diagram understanding has seen some traction in recent years as described in the aforementioned approaches, we are interested in producing a system that requires no prior information. To the best of our knowledge, there is no work of this nature in computer vision literature. Therefore, we can consider the prior work in diagram understanding by Kembhavi *et al.* [2] and Kim *et al.* [31] as inspiration, as we create our proposed system from scratch. Without being capable of using the AI2D dataset that systems of previous works were built off of, we are required to build datasets to train and evaluate our approach. Currently, there is no readily available datasets containing genetic models and ground truth for computer vision applications including image classification, object detection, and relationship parsing.

2.2 Technical Preliminaries

In this section, the technical preliminaries for computer vision applications used within the steps of the proposed solution will be discussed. These topics can be considered required knowledge for techniques used in this thesis.

2.2.1 Natural Imagery

Throughout computer vision research, natural images are the popular choice to be used while motivating and training deep neural networks. Training deep neural networks for computer vision applications is considered a data-dependent task. Applications of computer vision often rely on large datasets of natural images. These datasets often only contain a few main topics to summarize the image. As we know it in the modern-day, computer vision emerged following much interest in image recognition and more specifically image classification. Classically, the task of image classification requires a set of images that are labeled with a single category (class). These images are split into training and evaluation subsets. The network is trained on the training set, and the performance of the network is captured using the evaluation set. Often 20% of the available dataset is used as the evaluation set.

Before the introduction of deep neural networks classical computer vision often relied on low-level image features to solve computer vision tasks. The first large-scale natural image data was introduced in 1998, the Modified National Institute of Standards and Technology (MNIST) dataset [33]. The MNIST dataset contains a set of 70,000 single digit hand-written numeric imagery. The dataset was extended in 2017 by Cohen *et al.* [34] to 280,000 images.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [35] was released in 2010. The goal of this challenge was to evaluate vision image recognition algorithms on a standard benchmark. ImageNet also provided labeled images that can be used to train image recognition algorithms. This challenge showed the high potential of deep neural networks in computer vision. Furthermore, the interest that ILSVRC brought to these deep neural network approaches can be attributed to the rapid advancement in the field of computer vision in the past 10 years. ImageNet is an extremely large database designed for training visual mechanisms, specifically image recognition tasks. Containing more than 14 million images with approximately 20,000 categories and over 1 million images containing bounding box positions, ImageNet is one of the most widely used natural image datasets. A few example images from a selection of classes can be seen in Figure 2.5.

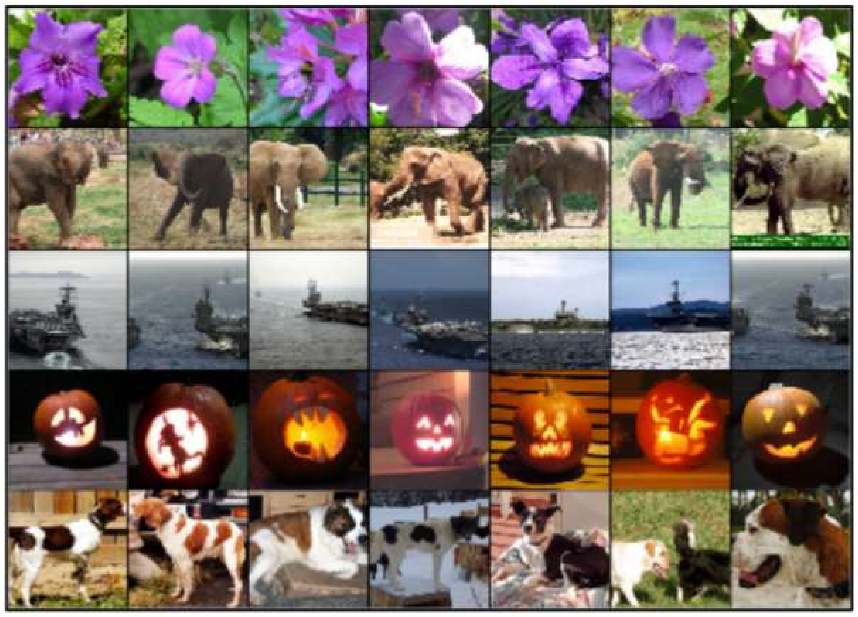


Figure 2.5. Examples of various image classes from ImageNet, including variations within the same class. Courtesy: Krizhevsky *et al.* [3]

Since the introduction of ILSVRC the error rate has continued to decrease annually. Performance of approaches in ILSVRC has come close to human-level accuracy recently. For the 2017 challenge, 29 of 38 competing teams achieved greater than 95% accuracy. ImageNet showcased the importance of data availability for training deep learning models.

As tasks within the field of computer vision continue to become more complex, image datasets need to evolve to meet the demand. More recent image datasets are an order of magnitude larger than ImageNet or MNIST. The Microsoft COCO (Common Objects in Context) [4] dataset is a large dataset that is primarily known for its wide assortment of available ground truth. Although expensive to collect this richly annotated dataset contains full scene segmentation as shown in Figure 2.6. Introduced in 2014, Microsoft COCO consists of over 328,000 images, with more than half containing complex annotations. Since its creation, Microsoft COCO has had a very diverse collection of use cases. For example, Microsoft COCO has been used in a knowledge base to create machine-developed structured sentences by training models that understand high-level concepts within the image [36].

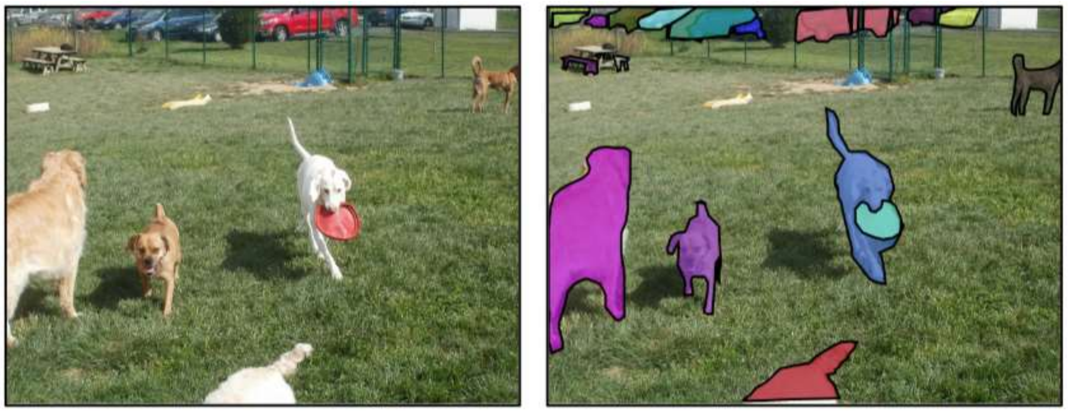


Figure 2.6. Original image (left) and the annotated image (right) from Microsoft COCO. Courtesy: Lin *et al.* [4]

Some of the available richly annotated ground truth in the Microsoft COCO dataset includes the following: object detection and instance segmentation bounding boxes; image caption; person instances labeled with keypoints; image segmentation masks; full scene segmentation including things (object) and stuff (background); and person instances labeled with key points.

2.2.2 Object Detection

Numerous deep neural networks have been proposed over the years for the task of image recognition. Image recognition can be considered a hypernym for computer visions tasks including object detection, image segmentation, localization, and image classification. Specifically, object detection is a fundamental challenge within computer vision works as it focuses on identifying and localizing objects present from known categories in the image. Furthermore, recent approaches to object detection [22, 32] explore combining various neural network layer types. Convolutional neural networks predominately saturate these architectures, yet each approach often proposes various tweaks between subsequent layers (activation, pooling, etc.). The application of deep learning to the task of object detection has seen human-level accuracy in recent works. Typically object detection assumes that both training and testing subsets are from the same domain and distribution of data.

Two popular approaches of object detection include: Single Shot Multi-Box Detector [32] and You Only Look Once [37, 38, 39, 22] and both have seen much success concerning precision and speed on natural image datasets [40, 41, 33, 4]. In this thesis, You Only Look Once (YOLO) has been chosen as the object detection network outlined in Section 5.1. Having similar performance to SSD, YOLO was selected based on its readily available code online. In the past few years, YOLO has seen multiple revisions which have iteratively increased its speed and performance. These versions include: YOLO9000 [37], YOLOv3 [38], YOLOv4 [39], and YOLOv5 [22].

The architectural similarity between all of these versions is the use of a triple pyramid network approach. This approach is used for classification using dimension clusters as anchor boxes. To train YOLO networks, a standard of Darknet formatting of labels which is [*class, centerX, centerY, width, height*] normalized between 0.0 and 1.0 to scale directly to the image. The architecture can be summarized into a two-stage detector which selects the valid resolution and highest confidence bounding box for a particular object in the image.

The deep learning object detectors often rely upon a feature extraction step to identify the class and location of an object. Feature extractors construct high dimensional representations of a given image. This portion of these deep neural networks is particularly important as even if object detection is not the final goal, these features can be used for other tasks on the same dataset. A handful of these popular deep neural networks in which the feature extractor's structure differs can be seen in Table 2.2.

Network Name	Authors	Year
R-CNN [42]	Girshick *et al.*	2013
VGG16 [43]	Simonyan and Zisserman	2014
Alexnet [3]	Krizhevsky	2014
Inception [44]	Szegedy *et al.*	2016
Resnet [45]	He *et al.*	2016
SqueezeNet [46]	Iandola *et al.*	2017
Densenet [47]	Huang *et al.*	2017

Table 2.2. Popular object detection networks in which their feature extractors have been seen in various tasks in computer vision.

The networks shown above in Table 2.2 have achieved the highest testing accuracies in the ImageNet Large-Scale Visual Recognition Challenge [35] over the past years. To obtain good performance on ILSVRC, the ImageNet dataset [40] is used to train these feature extraction networks. Furthermore, these feature extraction networks are publicly available with popular deep learning libraries such as PyTorch [48]. Publicly available pre-trained feature extraction networks are popular choices for new applications of computer vision in a variety of domains. Moreover, the mentioned approaches have been seen to create viable representations when applied to other computer vision tasks. Deep neural network approaches require training to create valuable representations making them need large datasets which may not be available in new target domains. Thus, image recognition approaches rely on hundreds or thousands of examples from each class to train a well-performing network.

2.2.3 Shared Weight Networks

One underlying assumption in deep learning is that deep hierarchical models which include CNNs, RNNs, and LSTMs create useful representations of the processed image in some high dimensional feature space. These extracted feature vectors from deep networks can be used to compare sets of images. This comparison began the work that uses shared weight networks for the purposes of image classification.

Siamese Neural Network

A Siamese neural network [49] takes two images as input and computes the distance (in some space) between these two images. Siamese neural networks can be used to decide if the two images belong to the same class. Feed-forward networks use shared parameters to ensure that the same network weights are applied to both input images. Ideally, when using a Siamese network two images of the same class should have little difference between their representations. Say S represents the network and $\mathbf{X}$ represents the dataset then

$$S(\mathbf{x}_1, \mathbf{x}_2) \in \{0, 1\}, \text{ where } (\mathbf{x}_1, \mathbf{x}_2) \in \mathbf{X}$$

these the two data items $\mathbf{x}_1$ (data item of interest) and $\mathbf{x}_2$ (reference class data item) are passed through a feature extraction network. These image feature vectors are passed through fully connected layers. The output of a Siamese network is a binary value that represents whether the image belongs to the same class (1) or not (0).

Triplet Network

A Triplet network [50] inspired by Koch *et al.* [49] is composed of three instances of the same feed-forward network. The approach of the Triplet network contrasts to a Siamese network which only has two instances. A Triplet network uses an L_2 distance comparison between an anchor to a positive and a negative example. Subsequently, the smaller distance is selected as the class for the anchor.

Say N represents a suitable feature extractor. N can be any one of the options listed in Table 2.2. $\mathbf{X}$ is a dataset of images. $\mathbf{x}_\mathrm{p}$ is a positive data item. This image belongs to the same class as the anchor. $\mathbf{x}_\mathrm{a}$ is the anchor data item. This image is our reference image which we are attempting to classify if its extracted features are closer to the positive or negative data item. $\mathbf{x}_\mathrm{n}$ is a negative data item. This image belongs to a different class than the anchor. The triplet set of images can be represented as

$$(\mathbf{x}_\mathrm{p}, \mathbf{x}_\mathrm{a}, \mathbf{x}_\mathrm{n}) \in \mathbf{X}$$

where given a set of three images ($\mathbf{x}_\mathrm{p}$, $\mathbf{x}_\mathrm{a}$, $\mathbf{x}_\mathrm{n}$) the goal is to determine which features extracted by N for $\mathbf{x}_\mathrm{p}$ and $\mathbf{x}_\mathrm{n}$ are most similar to $\mathbf{x}_\mathrm{a}$. We can assume that if the anchor image $\mathbf{x}_\mathrm{a}$ is closer to the positive image $\mathbf{x}_\mathrm{p}$, then $\mathbf{x}_\mathrm{a}$ belongs to the class of $\mathbf{x}_\mathrm{p}$. We compute the L_2 distance

$$T = \min(\|N(\mathbf{x}_\mathrm{a}) - N(\mathbf{x}_\mathrm{p})\|, \|N(\mathbf{x}_\mathrm{a}) - N(\mathbf{x}_\mathrm{n})\|)$$

between these features to discriminate the difference between the features, then select as the prediction the lower distance. This prediction is the resolving output of T, which would resolve the class that the anchor belongs to during evaluation.

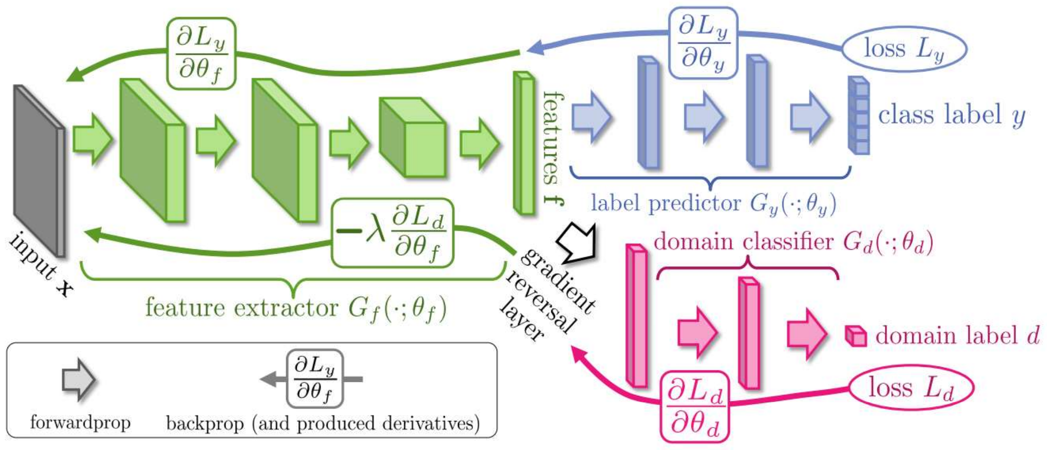
To train a Triplet network, Hoffer *et al.* [50] proposed Triplet loss. Triplet loss L_T

$$L_T = \max((\|N(\mathbf{x}_\mathrm{a}) - N(\mathbf{x}_\mathrm{p})\| - \|N(\mathbf{x}_\mathrm{a}) - N(\mathbf{x}_\mathrm{n})\|) + m, 0)$$

attempts to minimize the distance between the anchor and positive image over time while training. We compute the difference between the sets of (anchor, positive) and (anchor, negative), then additionally consider a margin m which is set with an empirically determined value before training. We define L_T as the following:

2.2.4 Domain Adaptation

The use of deep neural networks has been widely adapted across various niches of computer vision literature. Often the networks of these applications rely on large amounts



$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
features f
class label y
label predictor $G_y(\cdot;\theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
input x
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
feature extractor $G_f(\cdot;\theta_f)$
domain label d
$\frac{\partial L_d}{\partial \theta_d}$
loss L_d
forwardprop
backprop (and produced derivatives)

Figure 2.7. Proposed architecture for unsupervised domain adaptation. This architecture in particular focuses on the task of image classification. Courtesy: Ganin *et al.* [5]

of annotated data to tune weights for their respective task. Although some applications of computer vision do not have sufficient annotated datasets for supervised learning, methods using unsupervised training such as domain adaptation [5, 51, 52, 53, 54, 6, 29] can leverage the use of other datasets to apply a pre-trained network to a new domain of interest. These applications can still find other related datasets with viable training sets for a deep neural network, yet once the testing application is shifted to the desired domain for testing performance is not adequate.

Domain adaptation focuses on training a discriminative classifier that helps tune the weights trained on a source domain to improve the evaluation performance on a target domain. Domain adaptation differs from the widely used approach of transfer learning which applies a target domain's training set to a source domain's pre-trained weights. Primarily domain adaptation is useful for learning a mapping between two domains in which the target domain's dataset is not annotated (unsupervised domain adaptation) or has very few annotation samples (semi-supervised domain adaptation).

Typically when domain adaptation is used for a particular scope of imagery, annotated ground truth is either sparse or not available. Thus unsupervised domain adaptation can leverage learning domain invariant models in which both the image and annotation are available in the source domain while only the image is available in the target domain. Works within unsupervised domain adaptation in the scope of image classification mostly use distribution-matching-based approaches. These approaches ensure that features extracted from both domains are similar to each other using the maximum mean discrepancy or a domain classifier network.

The key contribution of domain adaptation is its use of a gradient reversal layer (GRL). The GRL acts as an identity transform to reverse the feature passing through the domain classifier. This has an impact on both the forward and backward propagation during training. The implementation of this layer into deep neural networks is trivial. Within the forward pass, it is an identity transform applied at the final feature vector, and in the backward pass, the gradient is multiplied by a constant λ. λ represents a scalar value which changes from 0 to 1 throughout training a given network. Further detail is included about domain adaptation and GRL below where a full example of domain adaptation is shown.

Domain adaptation assumes that there are no available labels for some target domain and a suitable amount of labels are available in the source domain. The lack of target domain data forces a given network to learn prediction features on the source domain dataset. Let us consider the following example:

Example of Domain Adaptation

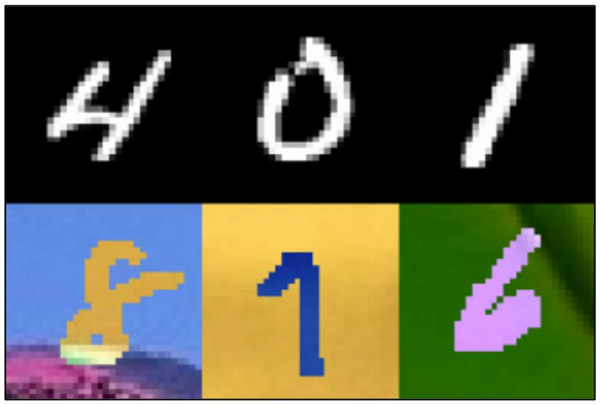


Figure 2.8. Datasets of use for desired domain adaptation application for image classification. Courtesy: Ganin *et al.* [5]

Given two datasets MNIST [33] (source domain) *S* and MNIST-M [5] (target domain) *T* as shown in Figure 2.8. The goal will be to train a classifier on images from the source domain. We can assume:

- *S*(**x**,**y**): Contains both imagery and labeled ground truth.
- *T*(**x**): Only has imagery.

To predict class labels as outlined in Figure 2.7, the source domain *S*(**x**, **y**) would be used in which $\mathbf{y} \in \mathbf{Y}$ of class labels. When using an image classification network on domain adaptation, this portion of the network's architecture is not modified, but the loss will now also consider the domain's loss. The primary change to the image classification network architecture will be the addition of the domain classifier and the use of a gradient reversal layer.

Predicting the domain label d_i of a given image in the source or target domain is a binary value. The domain of image $\mathbf{x}_\mathrm{i}$ domain is determined given some network *N*:

$$N(\mathbf{x}_\mathrm{i}) = \begin{cases} (\mathbf{x}_\mathrm{i} \in S(\mathbf{x}), & \text{if } d_\mathrm{i} = 0) \\ (\mathbf{x}_\mathrm{i} \in T(\mathbf{x}), & \text{if } d_\mathrm{i} = 1) \end{cases}$$

The goal of domain adaptation is to maximize the loss of the domain classifier while training the feature extractor of the network on the source domain. When back-propagating the network we have the loss for both the class label L_y and the domain label L_d. It is desirable to minimize L_y and maximize L_d. To maximize L_d, a gradient reversal layer (GRL) is used. The GRL which is inserted between the feature extractor and domain classifier requires no predefined parameters or hyperparameters besides the meta-parameter λ. Over the course of training λ is gradually changed from 0 to 1 which $p \in$ [0,1] is the training process. The change of λ can be defined as:

$$\lambda = \frac{2}{1 + exp(-10 \cdot p)} - 1$$

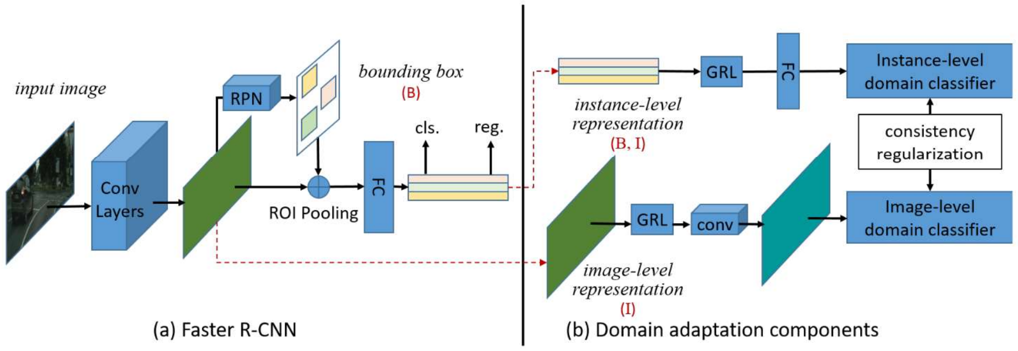

input image
Conv Layers
RPN
bounding box (B)
ROI Pooling
FC
cls.
reg.
(a) Faster R-CNN
GRL
FC
Instance-level domain classifier
instance-level representation (B, I)
consistency regularization
GRL
conv
Image-level domain classifier
image-level representation (I)
(b) Domain adaptation components

Figure 2.9. Domain Adaptive Faster R-CNN model, tackling the domain shift on both the image and instance level. Courtesy: Chen *et al.* [6]

During forward-propagation, the GRL acts as an identity transform using λ. Secondly, during the back-propagation, the GRL takes the gradient from the subsequent level, which is then multiplied by $-\lambda$ and passes it to the preceding layer.

Domain Adaptation on Object Detection

Domain adaptation has seen extensive applications on image classification tasks in computer vision as discussed in Section 2.2.4. These approaches are often used to adapt a given network from a large richly annotated natural image datasets [40, 41, 33, 4] (source) to a new dataset (target) which has little to no ground truth. Object detection networks perform well when evaluating on the same domain as the training set, yet shifts in the target dataset's distribution can significantly decrease performance.

Chen *et al.* [6] proposed an application of domain adaptation to be used on object detection, which is objectively more challenging as both the object location and category need to be predicted. The architecture shown in Figure 2.9 outlines their approach. Prior works using domain adaptation for image classification often only dealt with the class and domain loss. Object detection complicates the task of minimizing class loss and maximizing the domain loss as we are also interested in computing loss for the region and each instance level. Let L represent the summation of object detection loss such that:

$$L = L_{\text{det}} + \lambda(L_{\text{img}} + L_{\text{ins}} + L_{\text{cst}})$$

The proposed loss by Chen *et al.* [6] seen above also considers using a λ constant to tune the collection of domain losses. In this implementation, three losses are computed for the domain:

- L_{img}: Concerned with predicting the domain using the image-level representation being the tensor from the feature extractor before the region of interest and class is determined. The key benefit of predicting the domain using the feature map is to help reduce the shift caused by the fundamental image differences between domains.
- L_{ins}: Similar to L_{img} when predicting the domain, yet the features used are the instance-level representation. These features would be the flattened version of the image-level representation following their pass through various other steps of Faster R-CNN including the region of interest pooling and fully connected layers.
- L_{cst}: Maintaining consistency between the domain classifier on different levels which helps to improve the cross-domain robustness of bounding box prediction.

Inoue *et al.* [29] developed a solution for the task of object detection using domain adaptation with their cross-domain weakly-supervised approach. This is done by having instance-level annotation in the source domain, and image-level annotation in the target domain. Fully supervised detectors (FSDs) as described in Chapter 2.2.2 can be used to effectively capture features yet require large amounts of data to perform effectively. In this instance, FSDs are trained using the source domain and applied to target domains. Inoue *et al.* [29] proposes two methods which include:

- Using domain transfer to generate images that look like those in the target domain from the source domain, which has instance-level annotations.
- Using pseudo-labeling to generate pseudo-instance-level annotations. This is done by using an FSD on the source domain and applying it to the target domain. These candidates are processed into pseudo labels, furthermore, the FSD is then fine-tuned on the artificially generated samples using domain transfer as discussed above.

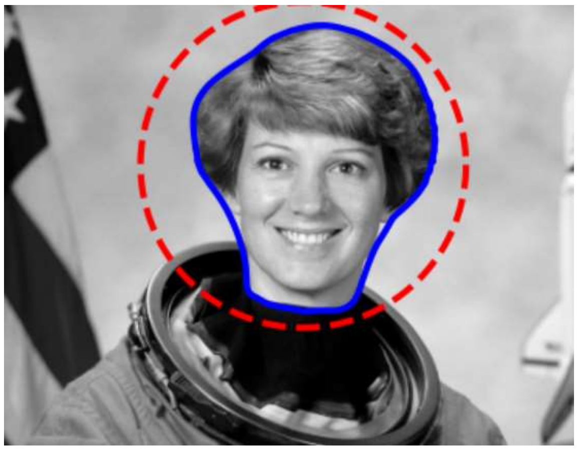


Figure 2.10. Example the Active Contour Model proposed by Kass *et al.* [7], an implementation used from scikit-image library. Red dashed line indicates the initial position, and the blue solid line indicates the final contour at the end of the routine. Courtesy: Scikit-image [55]

2.2.5 Active Contour Model

Although in modern computing the use of machine learning and deep neural networks seems to be the obvious choice to solve particular tasks, active contour models can be extremely useful for unsupervised line detection. In the case that labeled data is not present for a line detection task, active contours give the ability to find the pixel-wise position of the line from tip to tail. In 1988, Kass *et al.* [7] introduced an energy-minimizing spline guided by external constraint forces and influenced by image forces which cause the snake to converge on image features such as lines and edges. Active contour models [7] have practical uses for computer vision problems such as edge detection, line detection; contour completion, motion tracking, and stereo matching.

Computer vision research often relies on deep neural networks or blackbox approaches to find lines and edges. These autonomous approaches typically have well-developed interactive techniques for guiding these models to contours. Given an area of interest, as shown in Figure 2.10, there is an interest in converging the initial position of the algorithm's snake to the contours within this space. Using an energy-minimizing spline guided model [7] can solve this task. Throughout the active contour model task, the snake deforms itself from its initial state to conform to the nearest salient contour. Their procedure runs in Big Oh O(n) time complexity, for the iterative technique using sparse matrix methods.

Energy Functional

The external constraint forces are responsible for putting the snake near the desired local minimum. These forces are often predefined by the user which can be tuned to find particular contours, line, or edge styles. The active contour model's energy functional is defined as follows:

$$E^*_{\text{snake}} = \int_0^1 E_{\text{snake}}(v(s))ds$$

$$= \int_0^1 E_{\text{int}}(v(s)) + E_{\text{image}}(v(s)) + E_{\text{con}}(v(s))ds$$

We consider the energy functional of the snake as E^*_{snake} such that the position of the snake is represented by *v(s) = (x(s), y(s))*. Below each of the values within the energy functional will be explained:

- E_{int}: Internal Spline Energy – Considers a first and second order terms which changes over time as the snake's points approach corners.
- E_{image}: Image Energy – When dealing with the image energy functional. The three different energy functionals must be associated which attract a snake to lines, edges and terminations. The total image energy is expressed as a weighted combination such that:

$$E_{\text{image}} = w_{\text{line}}E_{\text{line}} + w_{\text{edge}}E_{\text{edge}} + w_{\text{term}}E_{\text{term}}$$

These further representations of *E* are defined as: E_{line} (Image Intensity), tuned given *I(x,y)* which will attract the snake to either light or dark lines; E_{edge} (Edge Functional), extends the E_{line} functional which will increase the snakes attraction to contours with larger image gradients often being corners or edges; and E_{term} (Termination Functional), used to find terminations of line segments and corners. Which is improved by increasing the smoothness of the image. Furthermore, this functional will attract to perpendicular gradients.

- E_{con}: External Constraint Energy – When initializing the snake, springs can be defined between positions x_1 and x_2. These springs are anchored at a fixed position and the snake is given movement with a constant k such that $-k(x_1 + x_2)$

Chapter 3 Datasets

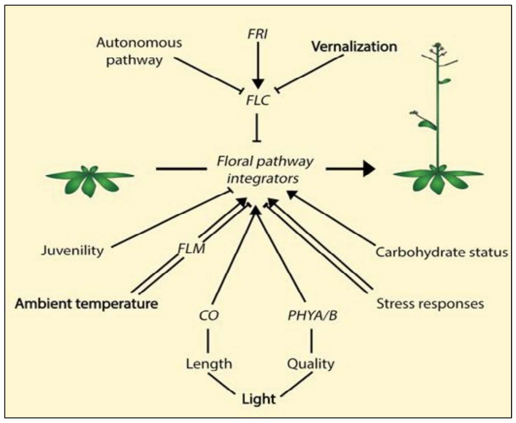
This chapter will discuss both the real and synthetic genetic model datasets titled GeneNet and GeneNetSyn, respectively. Training a deep neural network is a data-dependent task, thus we require large collections of data which is supplemented by richly annotated ground truth.

3.1 GeneNet Datasets

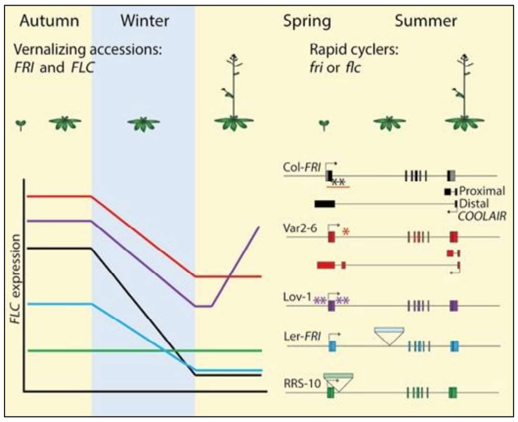
There have been many large-scale natural image datasets [40, 41, 33] available having with high-quality annotations for supervised tasks. Datasets containing thousands of examples for each class have been shown to train feature extraction models extremely well. Approaches that complete such tasks are trained to learn class-dependent features and distinguish inter-class differences to certain confidence. Deep neural networks trained on natural imagery are not suitable for systems that reason about visual illustrations. The challenges faced by such an approach to reason about genetic models is three-fold.

The first issue posed by publication figures is their high intra-class variance. In the natural image domain, classification datasets often contain low intra-class variances, meaning data items sampled from different image classes will not share the same visual characteristics. Having a low intra-class variance makes tasks of classification and localization easier for a given approach to leverage the differences to distinguish a data item's class. When dealing with publication figures with the goal of distinguishing if a figure is a genetic model or not, the high intra-class variance present makes this a non-trivial task. For example, if we compare two figures from the same citation which can be seen in Figure 3.1 where image *a* is a genetic model, while image *b* is not while both share very similar visual characteristics.

Secondly, throughout biology publication figures there is no standard for the styling and presentation of genetic models. For example, when considering a natural image object class such as a boat, we can assume that a large majority of imagery containing boats



a: Model



b: Non-Model

Figure 3.1. Comparison of the two GeneNet-98 classes. Images shown Bloomer *et al.* [8].

will have some of the visual characteristics such as a body of water, a beach, or a dock. Assumptions of this nature are understood by deep neural networks in a low-level way (pixel-wise). Generally, when a deep neural network approach is deciding if a given object is either a boat or car, it may consider pixel regions in proximity when developing a confidence metric. Comparing this example of boat imagery to publication figures, there is a wide variety of ways researchers will convey their contributions artistically including:

- Simplistic white background with black textual features and relationships.
- Collections of polygons with interior sets of genes with relationships among them. Additionally, relationships between each of the larger polygons connect each set of genes.
- One large background image such as a plant, leaf, genome visualization, etc. with the textual features and relationships of the genetic models contained within.
- Smaller sets of genes connected to one particular background image, created to look like a step-by-step growth pattern from left-to-right or top-to-bottom.
- Portions of the image containing a genetic model, and other portions including non-genetic models. An example of this could be an image of a plant or microscopic image which is "zoomed" in somewhere else in the image to describe a contribution via a genetic model.

- Similar colour palettes used on sets of textual features and relationships nearby which its purpose is only to look more visually appealing.
- Polygons containing sets of genes in which both the text font and background are very similar in colour.
- Textual features with and without background filling or stroke for every single element.
- Colours associated with a hierarchical layer (inwards or outwards change of gradient and colour scheme) for a set of genes, although this progression in colour is for visual appeal.
- Visualization of a biological system, the textual features of the genetic model are built into pieces of the background visualization.

As described, publication figures and genetic models can have extremely unexpected visual characteristics which makes applying deep neural networks extremely challenging. A few examples of these images that present challenges for a deep neural network are shown in Figure 3.2.

The last issue posed by publication figures is the amount of background clutter. This alludes to the previous point of unpredictable visual characteristics often included for artistic purposes. Typically, genetic models contain a lot of background noise or small constituents. These shapes containing noise often do not have any effect on the biological system presented, thus adding confusion to any given deep neural network. This background noise can also contribute to computer vision applications making errors when they rely on pixel-wise changes. Within our proposed system, we find this issue arises in the use of an active contour model in Section 5.2.

Currently, there are no publicly available publication figure datasets in the scope of plant biology. To train a computer vision system for diagram parsing, we require a richly annotated dataset of both publication figures and genetic models. We present the collection of GeneNet datasets that offer ground truth for genetic model classification, region detection, and diagram parsing. GeneNet introduces a new avenue for creating

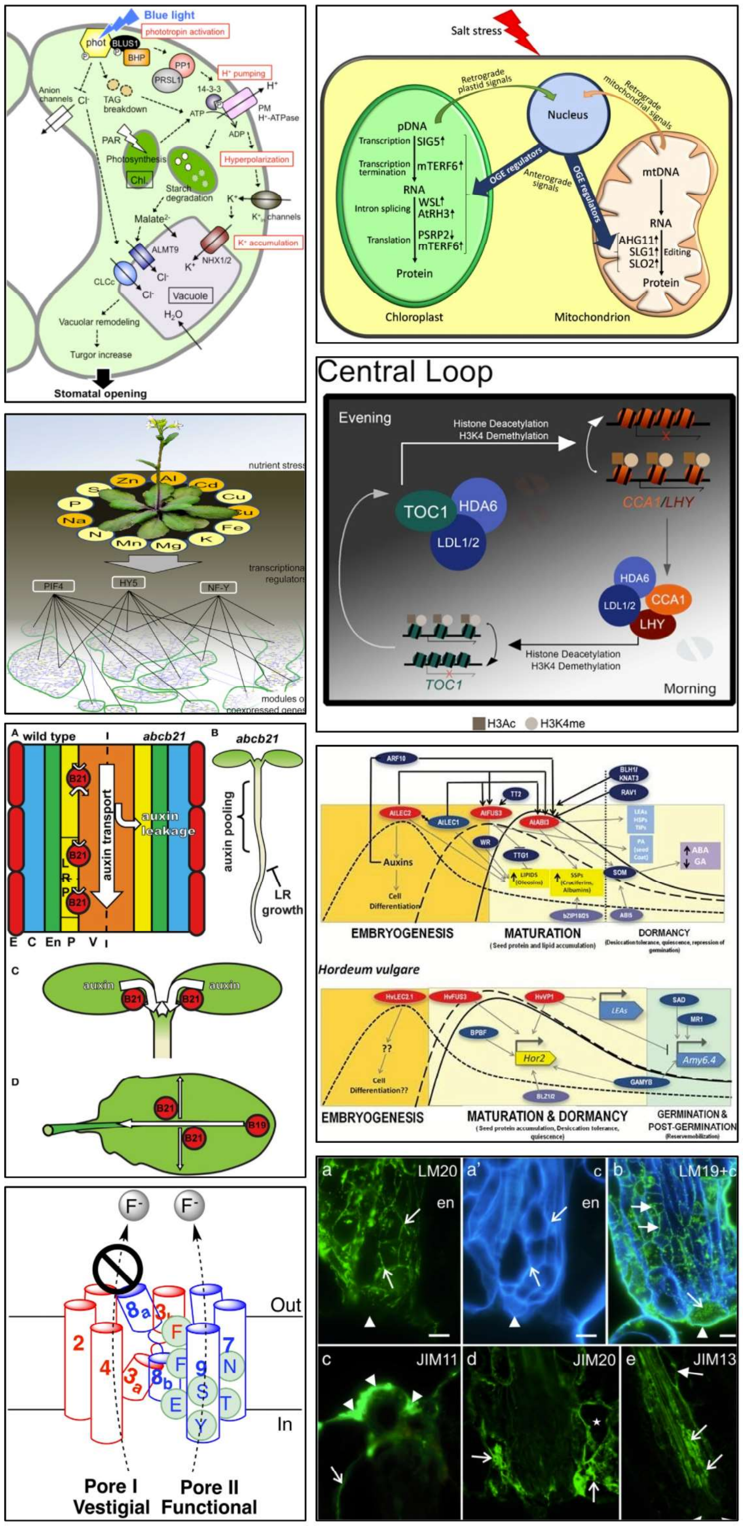


Figure 3.2. Collection of publication figures from GeneNet, including both genetic models and non-genetic models. Images shown are open access images from NCBI PubMed. Images shown courtesy: [9, 10, 11, 12].

Biological System	Abbreviation	Figure Count
Arabidopsis thaliana	AT	155,175
Hordeum vulgare	HT	10,428
Medicago truncatula	MT	5,960
Oryza sativa	OS	43,057
Physcomitrella patens	PP	2,620

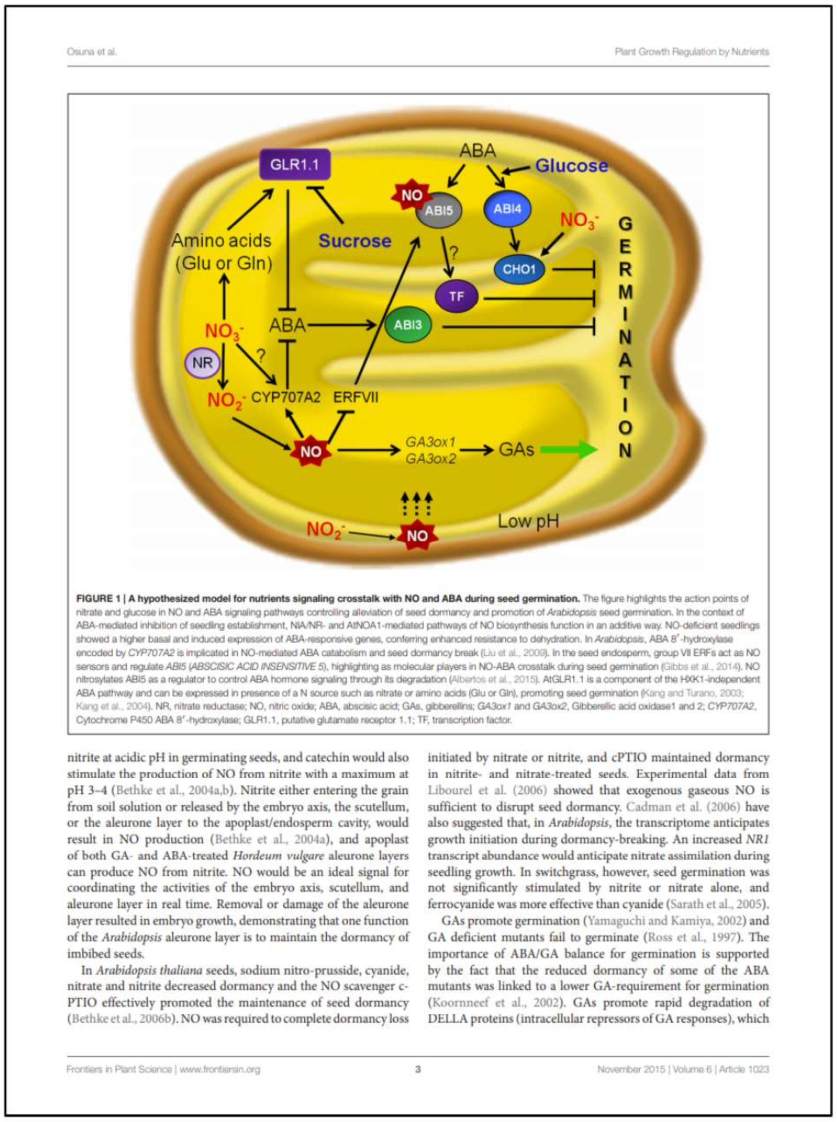
Table 3.1. Description of each biological system scrapped for GeneNet.

computer vision applications for genetic models analysis.

The National Center of Biotechnology Information (NCIB) PubMed comprises more than 32 million records of biomedical literature from MEDLINE, life science journals, and online books. The figures within these publications are of interest for the work within this thesis. To extract publication figures which will be filtered down to only genetic models we require a data scraping tool. Throughout this research we are interested in a few specific biological systems motivated by the Multinational Arabidopsis Steering Committee [27]. We focus our data collection of publication figures on the most researched plant biological system *Arabidopsis thaliana*. We are interested in collecting figures from citations that are open source and available on PubMed such as the page displayed in Figure 3.3. To gather these images, we use the Publication Figure Web Scraping tool developed by Alexander Sullivan from the University of Toronto's Provart Lab. This tool provides a method for scraping through NCBI's PubMed publications and retrieving the figures from open access and publicly available publications.

Although the work shown throughout this thesis contains specifically figures from the biological organism *Arabidopsis thaliana*, we also scrape the following organisms: *Hordeum vulgare*, *Medicago truncatula*, *Oryza sativa*, and *Physcomitrella patens*. From the aforementioned biological organisms, approximately 217,000 images were scraped which is broken down in Table 3.1.

From search queries based on the listed biological organisms, each is used as a search query to discover publicly available publications. An example of a publication figure extracted from a page in Osuna *et al.* [13] is shown in Figure 3.3. To ensure effec-



Osuna et al.

Plant Growth Regulation by Nutrients

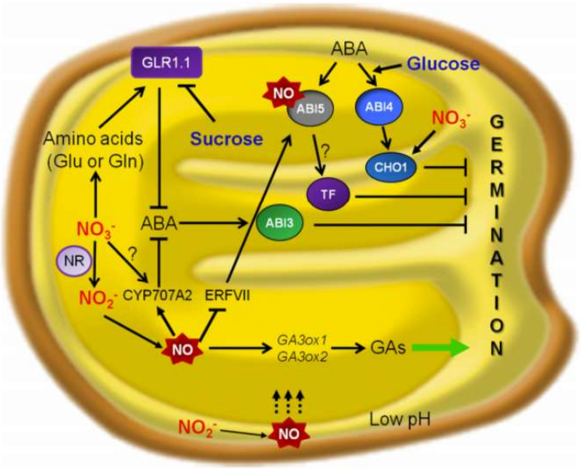


FIGURE 1 | A hypothesized model for nutrients signaling crosstalk with NO and ABA during seed germination. The figure highlights the action points of nitrate and glucose in NO and ABA signaling pathways controlling alleviation of seed dormancy and promotion of *Arabidopsis* seed germination. In the context of ABA-mediated inhibition of seedling establishment, NIA/NR- and AtNOA1-mediated pathways of NO biosynthesis function in an additive way. NO-deficient seedlings showed a higher basal and induced expression of ABA-responsive genes, conferring enhanced resistance to dehydration. In *Arabidopsis*, ABA 8′-hydroxylase encoded by *CYP707A2* is implicated in NO-mediated ABA catabolism and seed dormancy break (Liu et al., 2009). In the seed endosperm, group VII ERFs act as NO sensors and regulate *ABI5* (*ABSCISIC ACID INSENSITIVE 5*), highlighting as molecular players in NO-ABA crosstalk during seed germination (Gibbs et al., 2014). NO nitrosylates ABI5 as a regulator to control ABA hormone signaling through its degradation (Albertos et al., 2015). AtGLR1.1 is a component of the HXK1-independent ABA pathway and can be expressed in presence of a N source such as nitrate or amino acids (Glu or Gln), promoting seed germination (Kang and Turano, 2003; Kang et al., 2004). NR, nitrate reductase; NO, nitric oxide; ABA, abscisic acid; GAs, gibberellins; *GA3ox1* and *GA3ox2*, Gibberellic acid oxidase1 and 2; *CYP707A2*, Cytochrome P450 ABA 8′-hydroxylase; GLR1.1, putative glutamate receptor 1.1; TF, transcription factor.

nitrite at acidic pH in germinating seeds, and catechin would also stimulate the production of NO from nitrite with a maximum at pH 3–4 (Bethke et al., 2004a,b). Nitrite either entering the grain from soil solution or released by the embryo axis, the scutellum, or the aleurone layer to the apoplast/endosperm cavity, would result in NO production (Bethke et al., 2004a), and apoplast of both GA- and ABA-treated *Hordeum vulgare* aleurone layers can produce NO from nitrite. NO would be an ideal signal for coordinating the activities of the embryo axis, scutellum, and aleurone layer in real time. Removal or damage of the aleurone layer resulted in embryo growth, demonstrating that one function of the *Arabidopsis* aleurone layer is to maintain the dormancy of imbibed seeds.

In *Arabidopsis thaliana* seeds, sodium nitro-prusside, cyanide, nitrate and nitrite decreased dormancy and the NO scavenger c-PTIO effectively promoted the maintenance of seed dormancy (Bethke et al., 2006b). NO was required to complete dormancy loss initiated by nitrate or nitrite, and cPTIO maintained dormancy in nitrite- and nitrate-treated seeds. Experimental data from Libourel et al. (2006) showed that exogenous gaseous NO is sufficient to disrupt seed dormancy. Cadman et al. (2006) have also suggested that, in *Arabidopsis*, the transcriptome anticipates growth initiation during dormancy-breaking. An increased *NR1* transcript abundance would anticipate nitrate assimilation during seedling growth. In switchgrass, however, seed germination was not significantly stimulated by nitrite or nitrate alone, and ferrocyanide was more effective than cyanide (Sarath et al., 2005).

GAs promote germination (Yamaguchi and Kamiya, 2002) and GA deficient mutants fail to germinate (Ross et al., 1997). The importance of ABA/GA balance for germination is supported by the fact that the reduced dormancy of some of the ABA mutants was linked to a lower GA-requirement for germination (Koornneef et al., 2002). GAs promote rapid degradation of DELLA proteins (intracellular repressors of GA responses), which

Frontiers in Plant Science | www.frontiersin.org 3 November 2015 | Volume 6 | Article 1023

Figure 3.3. Page from the publication by Osuna *et al.* [13], we extract the figure shown which would then be parsed by our diagram parsing pipeline if it was predicted to be a genetic model.

tive collection of publication figures including both genetic models and other non-genetic model figures, a portion of the dataset was manually annotated with a binary classification for both positive and negative examples of a genetic model. Once we initially created these two sub-sets of this dataset on our own, Alexander Sullivan from the Provart Lab at the University of Toronto examined our annotations to ensure we correctly classified the genetic models.

Publications from the field of biology contain a wide variety of figures for various purposes throughout the article. Often being used to visualize a contribution to the biological system, genetic models were only found to be approximately 10% of the 155,175 publication figures scrapped from *Arabidopsis thaliana*. A large majority of publication figures are considered non-genetic models.

3.1.1 Classification Dataset

In particular, a filter was applied for only papers including *Arabidopsis thaliana* when initially creating the GeneNet datasets. A total of 155,175 *Arabidopsis thaliana* papers were filtered, unfortunately not every paper is open access so figures could not always be extracted. These publication figures can be split into two particular classes; genetic models and non genetic models: Models demonstrated in Figure 3.4 are figures of a genetic model map that contains a gene or a set of genes that biology researchers may be interested in searching for, and non-models which include publication figures not considered genetic models. These figures can include graphs, microscopic images, drawings, or natural images.

Papers written by various authors bring many challenges when creating a dataset to train a network to distinguish common features within an image. To create our classification dataset of biological diagrams containing both genetic models and non-models, approximately 7,500 images were manually annotated. The dataset was manually classified by ourselves, then validated by Alexander Sullivan in the Provart Lab at the University of

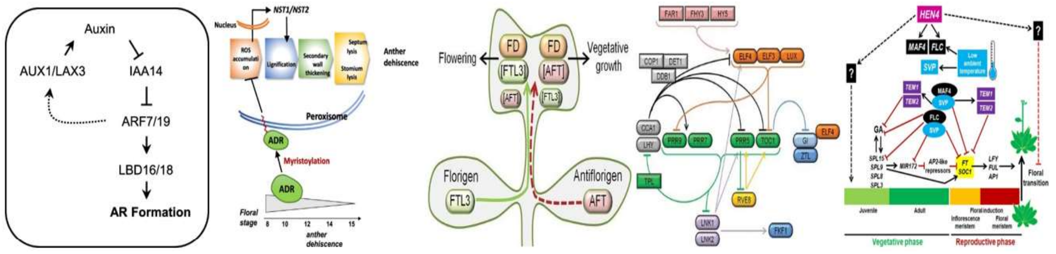


Figure 3.4. Example of the variety of genetic models within GeneNet Datasets. Genetic models shown courtesy: [14, 15, 16, 17].

Name	# Genetic Models	# Non-Models
GeneNet-98	55	43
GeneNet-500	208	292

Table 3.2. GeneNet Dataset versions, numerical value beside GeneNet representing the total number of images in the dataset.

Toronto. Throughout annotating the set of 7,500 images, it was found that genetic models account for a small fraction of the publication figures. The total number of classified genetic models and non-models were 208 and 7,292, respectively. Manually annotating these 7,500 images took approximately 40 hours. We present our two GeneNet datasets in Table 3.2, which outlines the total number of models and non-models within each and the suffix representing the number of images.

Being a small dataset compared to popular computer vision datasets [40, 41, 33, 34] this will create difficulty to train a network without over-fitting. Expanding the GeneNet dataset beyond its current size would require a large amount of time to manually classify, thus mechanisms using these datasets should consider deep learning techniques such as weight warm-up to reduce the network over-fitting to non-models. To maintain an unbiased dataset to avoid over-fitting in classification, a close to even distribution between models and non-models needed to be maintained. The figures extracted come in various height and width combinations from 200 to 1,500 containing square and rectangular shapes. The average size of images in the dataset was approximately 600 pixels in both the width and height. The average dimensions of the dataset are important, as when training a deep neural network all of the data items must be the same size.

3.1.2 Detection and Parsing Dataset

A desirable computer vision dataset contains various offerings of ground truth for each image. This can mean for one given image, ground truth is provided for a wide range of computer vision applications. We are interested in providing an additional subset of GeneNet which provides all of the ground truth required for validating a diagram parsing system.

We present our diagram parsing genetic model dataset named GeneNet-DP. GeneNet-DP contains 18 manually annotated genetic models which were selected from our set of diagrams in GeneNet-500. These genetic models were manually annotated by ourselves, which is an extremely time-consuming task. Listed below, each item within the dataset will contain all of the ground truth to ensure effective validation can be collected for both object detection and diagram parsing tasks.

- Manually annotated bounding boxes for object detection purposes.
- Textual feature identification and labelling to each associated bounding box.
- Triplet sets of relationships (entity, relationship, entity) for diagram parsing.

To manually label our object detection bounding boxes, we use a publicly available Python script named LabelImg which is shown in Figure 3.5. During this annotation, the textual features within blobs were manually paired with the labeled bounding box in a sequential nature. Specifically, we validate the performance of our final result in the proposed diagram parsing algorithm discussed in Chapter 5.2. In Appendix B we present a full data item from our GeneNet-DP dataset.

3.2 Synthetic Datasets

In this section, the development and details of all versions of synthetic data are discussed. We propose a collection of datasets that iteratively increase in visual complexity. The goal of these synthetic datasets is to attempt to mimic the interaction of genes through relationships present in genetic models from publication figures present in GeneNet.

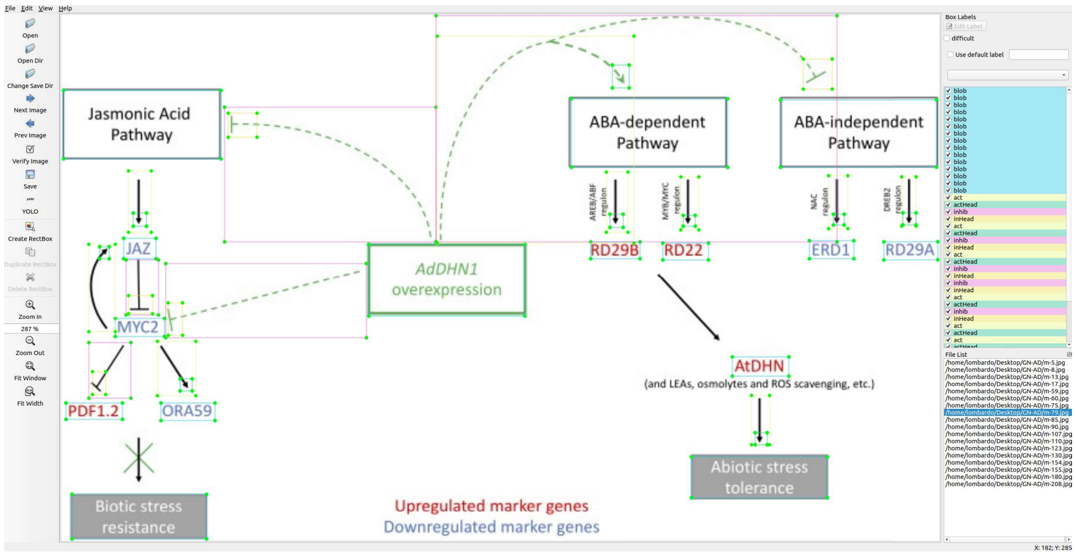


Figure 3.5. Demonstration of the annotation tool (LabelImg) used for GeneNet-DP. This public tool can be found at: ***https://github.com/tzutalin/labelImg#installation***

Developing a system for diagram parsing described in Section 1.3 requires an assortment of computer vision applications. These applications are mostly deep neural networks which are highly data-dependent, which means sufficient training data is required. Currently, the limitation of the work proposed in this thesis is the lack of richly annotated data. One of the popular visual illustration datasets proposed by Kembhavi *et al.* [2] is the AI2D dataset which is a grade-school science diagrams. Although a richly annotated visual illustration dataset, there are no plant biology publication figures or genetic models present in the dataset thus it is not sufficient for our work. We introduce GeneNetSyn, our collection of synthetic genetic model datasets. GeneNetSyn can be broken into four versions, each version has one major contribution to the generation or structure of the genetic models. The versions of GeneNetSyn are summarized below:

3.2.1 Version 1

Developing a dataset of richly annotated biological diagrams is not only expensive but time-consuming. Developing an automated system that creates synthetic diagrams can expedite the process of manually annotating a large set of real biological diagrams.

Version #	Contribution Description
1	Collection of ten variations increasing in complexity. Randomly generated positions for blobs. Short sequences of text.
2	Structurally generated positions for blobs. Longer textual sequences dependent on blob width.
3	Increased number of possible blobs within image. Equal chance for activation, inhibition and false lines. Chance for rotated textual sequences and multi-line sequences.
4	Experimental dataset excluded from evaluation. Use of copyright-free biology imagery as the background of the genetic model.

Table 3.3. Summarization of each version within GeneNetSyn.

GeneNetSyn Version 1 contains ten variations of the dataset which increase in complexity. We propose these ten variations in Version 1 to ensure the validity of the synthetic diagrams as we attempt to replicate genetic models in Section 3.1. These variations shown in Table 3.4 each offer a significant change as Version 1 develops. The development ranges from images with four simple circular blobs to up to 10 blobs with varying sizes and a wide range of relationship representations.

The aforementioned variations of GeneNetSyn Version 1 contain 12,000 images which are broken into a 10:1:1 split being training, validation, and testing respectively. The variations of Version 1 all follow the same random blob position generation approach. The random blob position generated relies on each subsequent blob position and a fixed minimum distance of 35 pixels between blobs. These datasets all come with ground truth and can be used for future works including: Darknet format normalized ground truth labels; Raw ground truth of bounding boxes; Number of blobs and relationships; Text within each blob; and Triple (entity, relationship, entity) sets for each relationship, offering both (text, type, text) or (index, type, index). An example of a diagram from each variation is shown in Figure 3.6.

Index	Name	Max Blob #	Description
1	Circular	4	Circular blobs only
2	Text Only	8	Textual blobs only
3	Circular Text	8	Mixture of circular with text and only textual blobs
4	V Blob Circular	8	Mixture of circular and textual blobs
5	V Shape V Text	8	Blobs are random shaped ellipse and rectangles
6	V Curves	8	Relationship lines can be a randomly seeded Bezier curve. Variable arrow colour.
7	V Dashes	8	Relationship lines can be dashed
8	V Noise	8	Random shapes added to the background. More fonts added
9	V False Lines	10	Relationship lines present without head type association.
10	V Heads	10	Relationship head types added to classes

Table 3.4. Variations of GeneNetSyn datasets. V denotes Variable. Contributions are carried downwards throughout the variations.

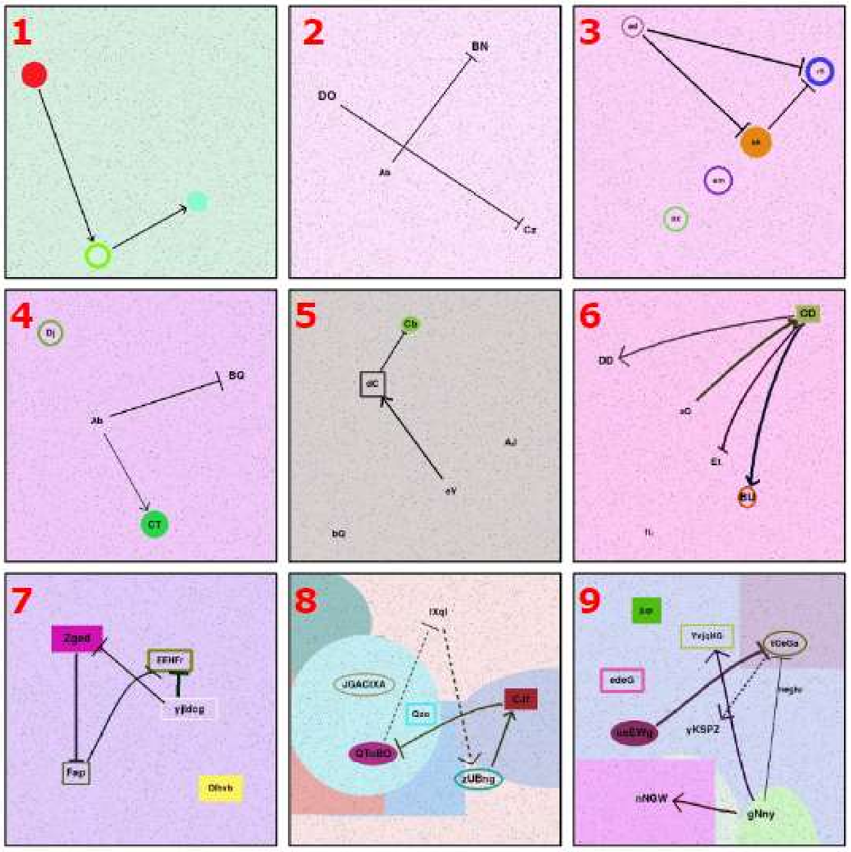


Figure 3.6. Example diagrams from each GeneNetSyn variation. Red indexing at the top-left of each image represents the variation number from Table 3.4.

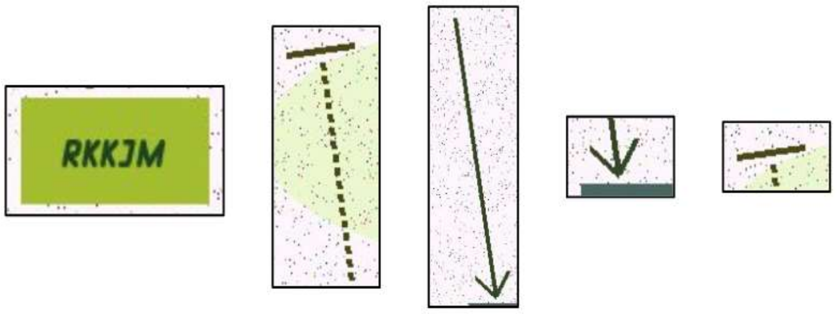


Figure 3.7. Visualization of the 5 object classes within GeneNetSyn; blob, inhibition, activation, act-head, and in-head. Shown from left to right respectively.

The underlying assumption of a genetic model is that there will be entities that are bound together through a given relationship. To develop a synthetic diagram of a genetic model, we require five classes:

- **Entity**: Any gene or gene product constituent in the diagram, often represented as text paired with a shape surrounding the text. In some biological diagrams, these entities may just be text.
- **Activation**: Flat tipped line relationship between two entities.
- **Inhibition**: Arrow tipped line relationship between two entities.
- **Activation Head**: Flat tipped line arrow head.
- **Inhibition Head**: Arrow tipped line arrow head.

In the later variations within Version 1 and beyond, we additionally add the classes of the head type. Relationship head location as an independent entity is important to allow us to isolate the tip and tail of the arrow. This isolation will be extremely useful when attempting to predict the direction of a given relationship between two blobs. These image classes are shown in Figure 3.7.

The aforementioned object classes appear in a wide range of shapes and styles as diversity over the synthetic datasets. Blobs formally considered textual features are the most important object class to locate when dealing with genetic models. Blobs require a vast amount of visual diversity which includes: font sizes, font types, textual element length, background object, background shape, and background hollowness. Both inhibition and activation relationships fall into the same category of visual diversity as their characteristics are generated before selecting which relationship is present. The visual characteristics of inhibition and activation relationships include length from blobs, line thickness, solid/-dashed line, curvature, and direction.

Within biology and bioinformatics publications there is no set of rules to standardize the style, colour, or background clutter in a publication figure. This lack of standardization makes these images quite difficult to replicate. To understand some of the global characteristics of GeneNetSyn Version 1, Figure 3.8 offers four visualizations of distributions within the variable noise dataset (index 8).

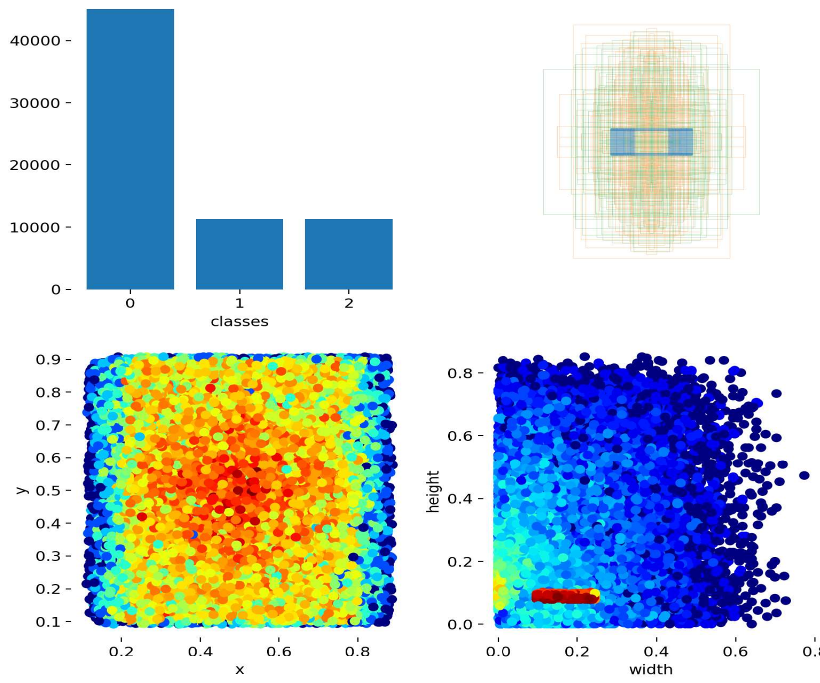


Figure 3.8. Visualizations of variable noise dataset (Index 8) showing: object class occurrences counts; possible bounding boxes; location of center points for each class item; and the width and height for all object classes. Visualizations described are shown clockwise from top-left. Scale using the RGB jet colour map (blue weak, green moderate, and red strong intensity)

3.2.2 Generating Images

To generate these images Pygame was used to draw the images, as briefly discussed in Appendix A.2. Images in the dataset are a fixed canvas size of 512 pixels with both the height and width dimensions. The datasets contain a random number of blobs *N* and relationships (1 - *N-1*) to a maximum of *N*=10 in Version 1. Each given diagram is built from the ground up, such that we begin with a random background colour and progressively add detail and complexity to the diagram.

Given a particular generated image, there are two primary variables that help to develop the image which is: maximum constituent count, a minimum distance between blobs which were set to 10 and 40 respectively.

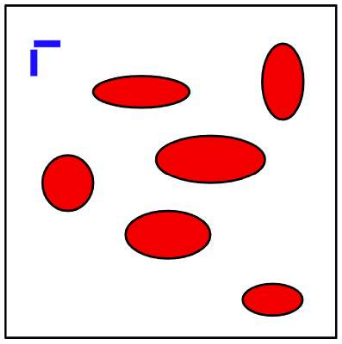


Figure 3.9. An example of some blobs generated by the Version 1 random position approach. Red ovals indicating blob regions, and the blue lines signifying the minimum distance between blobs.

Version 1 of GeneNetSyn takes an approach of randomly generating blob positions. This random generation is bounded by two factors: other previously generated blob location plus the minimum distance, and the location to the edges of the canvas plus or minus the given scale of the current blob being generated.

Publication figures found in GeneNet, more specifically the genetic models have inherent distance between blobs with a relationship in-between them. Fundamentally a minimum distance has to be present between two blobs with a relationship to be capable of visualizing the relationship line and head type. Additionally, blobs cannot share multiple relationships between the same blob. An example can be that blob *A* cannot have both an inhibition and activation relationship to blob *B*. There is no standardization for the spatial location of relationships which can be assumed as order (vertical-based hierarchy as an example), therefore relationship locations are completely random.

We first began by ensuring that our object detection network is capable of locating and identifying identities in images from GeneNetSyn Version 1, row 1 in Table 3.4. The addition of textual features is considered the most important part of genetic models as we require two textual elements to connect together when inferring relationships. The textual feature within a given blob is randomly generated for the variations of GeneNetSyn Version 1 of the datasets. In the first six datasets (i.e; first 6 rows in Table 3.4) text is generated in an ordered fashion from A to Z as the prefix which is paired with an additional random letter. The ordered prefix of textual features was essential initially to qualitatively visualize where blobs were being randomly generated. The initial datasets were limited to only textual features of length two to ensure the detection network was capable of generalizing various visual characteristics of blobs before their increase in length. Later datasets in GeneNetSyn Version 1 had textual feature-length dependent on the given width of the current blob. This amount of characters *C* present in a textual feature was computed by using the current blob width *BW* and the current font size *FS*.

$$C = \left\lfloor \frac{BW}{FS} \right\rfloor$$

One important feature of synthetic diagrams which is important to challenge a neural network to properly learn how to identify classes is the use of noise. Each image is completely random concerning the blob and relationship locations, blob colour, background colour, and background noise (1/30 chance per pixel). The background noise chance of 1/30 per pixel was empirically determined through analyzing imagery qualitatively to find a good balance of noise without fully saturating the initial background colour. After the introduction of variable noise (row 8 shown in Table 3.4), background shapes were added to attempt to confuse the neural network. Often in real biological diagrams background shapes are added for an artistic impact.

To add more complexity, there is no restriction on where relationships can occur. The lack of restriction causes a good portion of images to have relationships overlapping with blobs and relationships. These can have overlapping relationships which lead to false-positive errors for systems attempting to infer relationships, resulting in the addition of the aforementioned restriction in later versions of GeneNetSyn.

A genetic model contains both inhibition and activation relationships throughout the diagram. Fundamentally both relationship types contain a line between two blobs contained within the relationship. Relationships are determined based on the head type at the tip of the line. When generating both relationship types, a different approach is required for how the relationship head is generated:

Inhibition

Inhibition relationships are recognized by a second perpendicular line at the tip of the relationship line. Given the two points of a candidate line (tip and tail), we need to generate the perpendicular points for the tip to create our inhibition relationship head. To do this, we obtain the left and right parallel offset which are the two (x,y) positions the line segment formed connecting these two points is perpendicular to the line segment terminating at the tip of the relationship curve. Lastly, we simply need to determine the final length of both sides of the inhibition relationship so its scale matches closely to the relationship. Given the width and height of the full relationship, we use the average value multiplied by an empirically determined 1.4 value to compute the actual length of both inhibition line sides. This value of 1.4 was the largest value that the inhibition lines could be multiplied by to not span outside of the original relationship bounding box.

Activation

Activation relationships are recognized as an arrow at the tip of the relationship line. Given the two points of a relationship line (tip and tail), we need to generate the two additional points to create the arrow at the tip of the line. Thus, we must compute the vectors from the tip to tail and vise versa in the *x* and *y* coordinate system. Following these vectors being computed, we then use the norm of both aforementioned vectors and some randomly generated amount of noise to generate the *x* and *y* points for both arrow tips. We need to use the norm in both directions as we need to assume that arrows can be in any orientation, not simply on each right angle or diagonal.

A full example of a data item from GeneNetSyn can be found in Appendix B. Some example images from GeneNetSyn Version 1 sampled from all of the variations can be seen below in Figure 3.10.

3.3 Version 2 & 3

Following the completion of the GeneNetSyn Version 1 datasets, the approach to generating these synthetic diagrams was overhauled. The approach overhaul was primarily focused on the spatial location of blobs throughout the diagram. It was found that as the number of constituents within Version 1 datasets increased, the number of false positives and overlapping relationships also increased. Furthermore when the larger textual features were introduced which was dependent on blob width another issue arose. Object instances tended to converge towards the middle of the canvas, although being initialized on corners. This limitation caused a reduction in the number of possible blobs which could be generated in images, due to the required minimum distance from previously existing blobs. To visualize the previously stated drawbacks and proposed solution, two variations of GeneNetSyn Version 1 accompanied with Version 2 and 3 are shown in Figure 3.11.

One key difference between Version 1 to both Version 2 & 3 is the number of maximum constituents *N*, which was increased to 12. Although in the figure shown above it seems that GeneNetSyn Version 1 variable dashes have more spatial diversity, although its average blob count is 5.3 compared to Version 2 & 3's 8.6. Increasing the number of blobs in an image not only increases the difficulty of inferring the relationships present

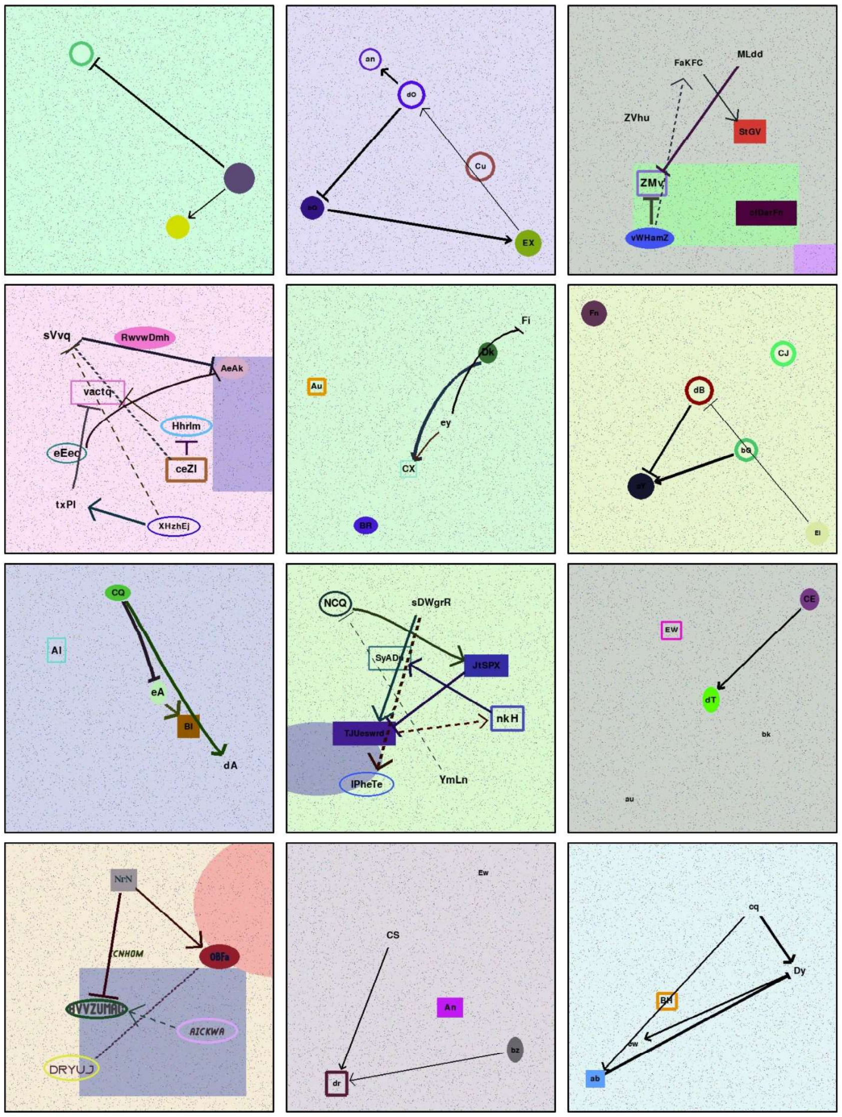


Figure 3.10. Mixture of example images sampled from the variations in GeneNet-Syn Version 1.

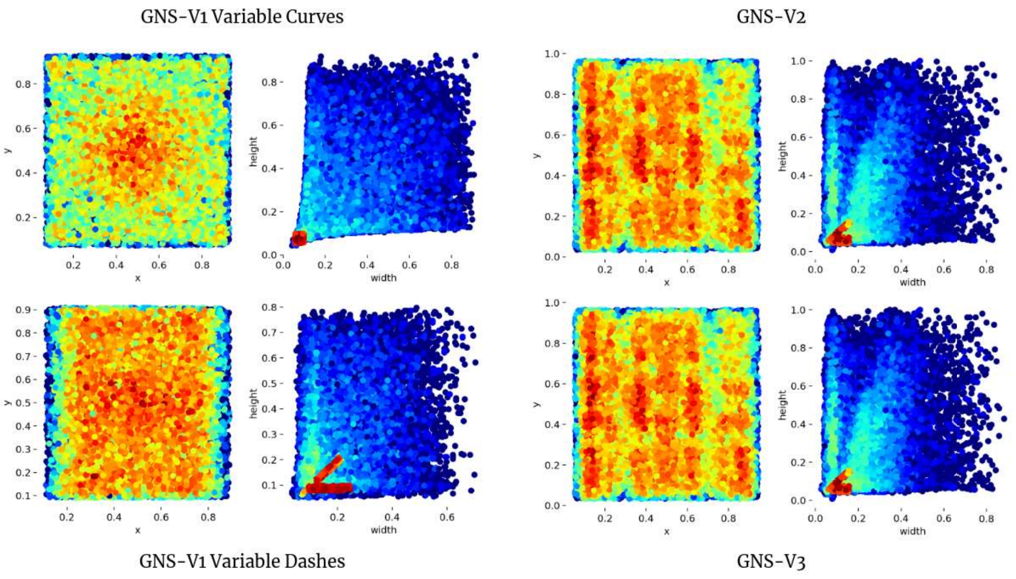


Figure 3.11. Visualization of two variations of GeneNetSyn Version 1, GeneNetSyn Version 2, and GeneNetSyn Version 3. Displaying the center origin of object instances (left sub-image) and the width & height of object instances (right sub-image). Scale using the RGB jet colour map (blue weak, green moderate, and red strong intensity)

but will increase the number of objects to be detected while training a deep neural network approach. The proposed approach to generating blob positions is a grid assignment based on the initial number of blobs to be generated. This approach contrasts to what was used in Version 1 which relies on a randomly generated position constrained by other blobs in the image. Our change is visualized in Figure 3.12, in which we first generate the total number of blobs for the diagram then assign a region of space each blob can reside. Furthermore, this new approach will reduce the amount of noise for a relationship's line, as previously it would cross through other blobs in the image thus throwing off the understanding mechanism and region detection.

GeneNetSyn Version 2 introduces this new approach, and takes the highest complexity datasets from Version 1 and combines them all. These key features that we combine include dashes lines, false lines, longer textual features, curved lines, and varying background noise.

Continuing to increase the difficulty for deep neural networks, Our Version 3 of

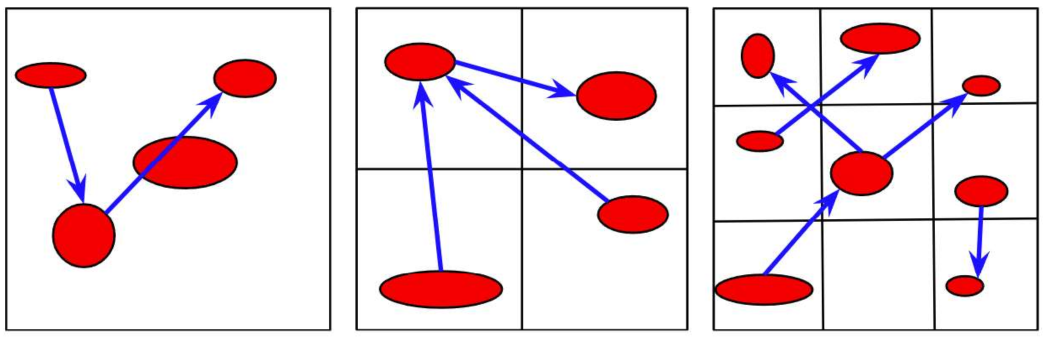


Figure 3.12. Comparison of blob and relationship generation pattern between GeneNetSyn Version 1 (left) and Version 2 & 3 (center and right). Specifically the left and center image are showing the difference between a random and procedural approach to selecting 4 blob positions.

synthetic diagrams goal is to increase the difficulty of finding groups of textual constituents. This can be conveyed in two separate ways. Firstly, we can introduce angled text by tilting the textual features. Secondly, there can be blobs that contain multiple lines of text. In real biological diagrams, a collection of genes are considered a group that only one relationship connects to. Thus to effectively attempt to closely replicate real biological diagrams we need to increase the diversity of text.

3.3.1 Experimental Version 4

One of the key characteristics which challenge approaches on publication figures from works in biology such as our genetic models discussed in Chapter 3.1 is background imagery. Background imagery in publication figures makes it extremely difficult for computer vision approaches to reason about the given genetic model as background information can easily be confused with regions of interest. We present GeneNetSyn Version 4 which we offer as an experimental dataset for motivation of future work. To develop this Version of GeneNetSyn, we collect a sample of copyright-free images following search queries on Google Images including “biology”, “plant system”, and “genetics”. We consider GeneNetSyn Version 4 an experimental dataset to provide insight of our future works as we continue to increase the complexity of genetic models. Thus, GeneNetSyn Version

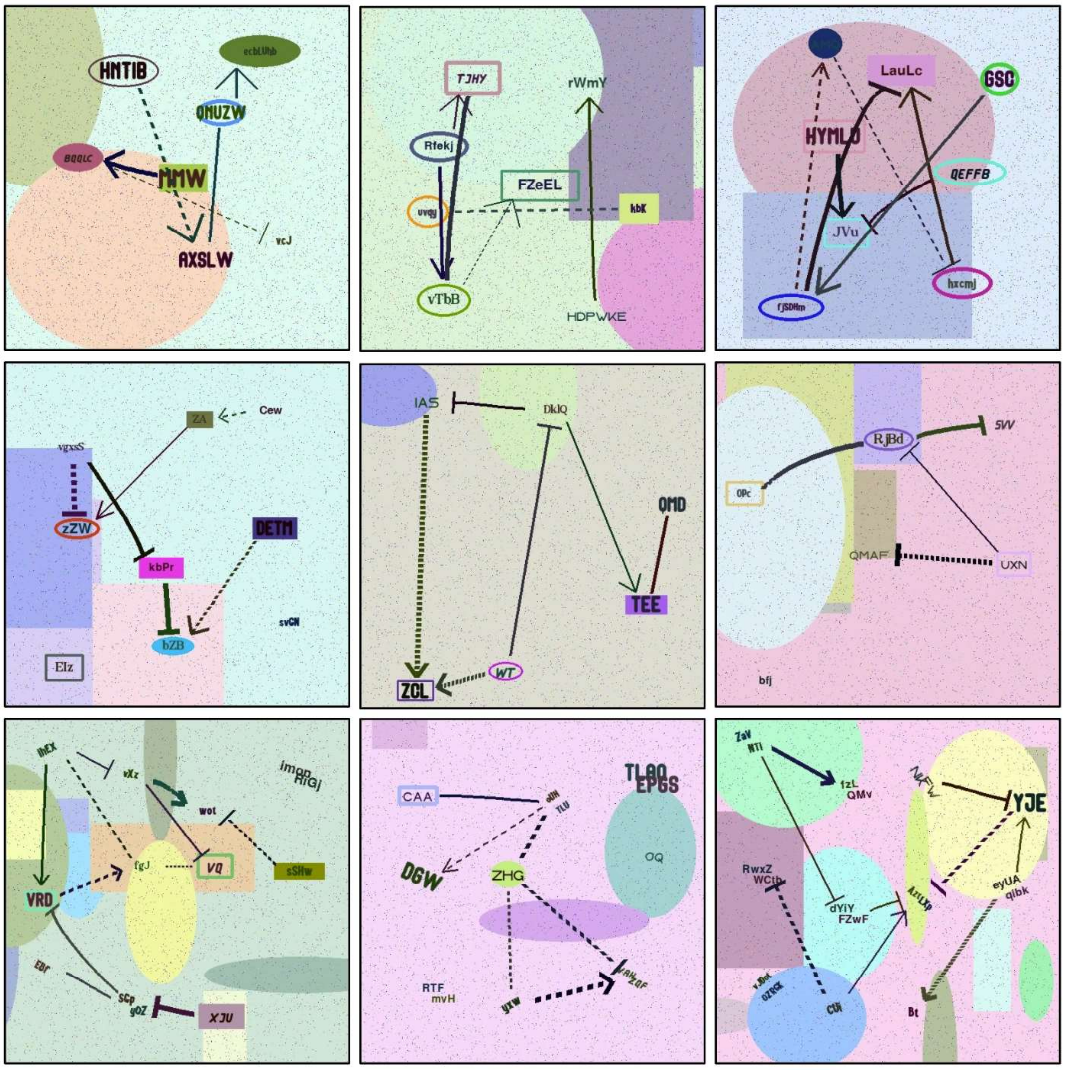


Figure 3.13. Visualization showing the key visual changes between the first 3 versions of GeneNetSyn. GeneNetSyn Version 1 (top diagrams), Version 2 (middle diagrams) and Version 3 (bottom diagrams).

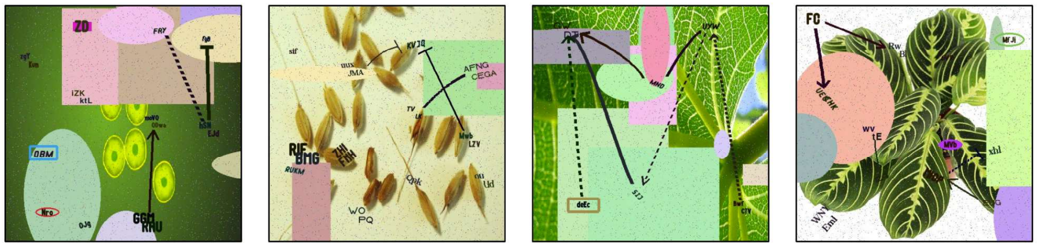


Figure 3.14. Demonstration of copyright-free images incorporated into synthetic genetic models. An experimental Version 4 of GeneNetSyn.

4 was not included in any evaluation of our proposed system.

Chapter 4 GMM Classification

The first pillar of computer vision applications has the goal of taking a given publication figure and determining if it is a genetic model. This section introduces a lightweight feature extraction network specifically for genetic models titled GeneModelNet (GMN) whose convolutional architecture is inspired by VGG19, and AlexNet. We apply the extracted feature maps of GMN, VGG19, AlexNet, and SqueezeNet 1.1 to a Triplet classification network to determine if a given diagram is a genetic model. The dataset used to train and validate our approaches is GeneNet-98 and GeneNet-500, our manually annotated dataset of genetic models and other publication figures introduced in Section 3.1.

4.1 GeneModelNet

GeneModelNet (GMN) inspired by VGG [43] and AlexNet [3], contains six learned layers including four 2D convolutions and two fully-connected layers. GMN is introduced to evaluate the performance of a feature extraction network with initialized uniform weights. GMN is used to compare how effective a feature representation is developed by its approach to pre-trained feature extraction models on natural imagery. GeneModelNet's convolutional layers use varying kernel sizes from 8×8 to 4×4 with a stride of 1 and padding of 0. Following each convolutional layer, the output passes through a ReLU activation, then a 3×3 max-pooling. Given the large image input size of 600×600, a larger max-pooling size was chosen to help reduce the dimensionality following each of the convolutional layers. Following the last convolutional layer, the features are flattened and a dropout is applied probability p is set to 0.2 (20%).

The base architecture of GeneModelNet's learned layers is maintained in the two variations; GMN144 and GMN1028, where the numerical suffix signifies the final feature output size. The aforementioned variations of GMN differ based on the selection of kernel sizes, which affects the rate at which the model feature is reduced. Through initial experimentation of performance when developing the two architectures, it was determined that when GMN contained more than four convolutional layers its performance began to decrease. These experiments evaluated a wide variation of convolutional neural network layers, where four layers were found to be the ceiling of performance. Therefore the total number of convolution layers used in both GMN144 and GMN1028 was four. GMN is trained throughout the training of the Triplet classification network outlined in Section 4.2. Initially containing uniform weights, we use the loss computed from the Triplet classification to tune the weights of GMN.

4.2 Classification Network

We introduce our approach to determining whether a given publication figure is a genetic model or not. A Triplet classification network shown in Figure 4.1 is leveraged to complete this task. A Triplet network [50] discussed in Section 2.2.3 requires three input images to decide whether the image of interest (anchor) is more similar to the positive or negative class from the dataset $\mathbf{X}$. For each data item passed to the network, the data is divided into three sub-batches: the image of interest (anchor) $\mathbf{x}_{\mathrm{a}}$, the positive class example $\mathbf{x}_{\mathrm{p}}$, and the negative class example $\mathbf{x}_{\mathrm{n}}$).

$$(\mathbf{x}_{\mathrm{p}}, \mathbf{x}_{\mathrm{a}}, \mathbf{x}_{\mathrm{n}}) \in \mathbf{X}$$

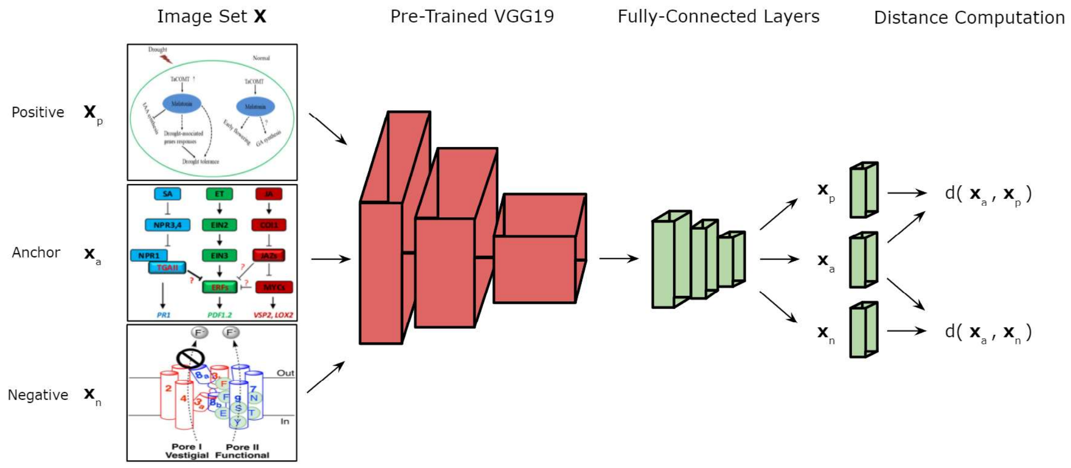


Figure 4.1. Breakdown of the Triplet classification network used for genetic models. The proposed Triplet classification network can be separated into two steps, which are image feature extraction and distance metrics. Fully connected layers depicted are 3,600 to 1,800 to 10. Images shown are open access images from NCBI PubMed.

To obtain the best performing network, we must evaluate how an assortment of feature extraction networks create image features on genetic models. The feature extraction networks used include: VGG19 [43], AlexNet [3], SqueezeNet 1.1 [46], and GMN. Besides GMN, the three feature extraction networks selected are popular approaches used for various image recognition tasks. Each of these approaches architecturally is different, yet produces similar output feature vectors before the image classification layers of their networks. The aforementioned feature extraction approaches are trained on ImageNet [40] for the ImageNet Large Scale Visual Recognition Challenge [35]. Although trained on natural imagery, these pre-trained models can be used in other domains. The given assumption is that these vigorously trained models can still produce a feature vector that represents the image effectively for classification. Each of the three input data items ($\mathbf{x}_p$, $\mathbf{x}_a$, $\mathbf{x}_n$) are passed through these feature extraction networks separately on the same shared weights.

Following feature extraction of ($\mathbf{x}_p$, $\mathbf{x}_a$, $\mathbf{x}_n$), we require classification layers to identify which image class each image belongs to. The image classification layers of VGG, AlexNet, and SqueezeNet are removed and replaced with two fully-connected layers trained by the loss computed by the Triplet network. After being flattened, the fully-connected layers reduce the dimensionality of the feature extraction model from 3,600 to 1,800 to 10. This feature vector of 10 values is trained to represent if the processed image belongs to either the genetic model or non-genetic model class. The resulting output feature vector has a Sigmoid function applied which gives the final feature vector for each data item ($\mathbf{x}_\mathrm{p}$, $\mathbf{x}_\mathrm{a}$, $\mathbf{x}_\mathrm{n}$).

The final feature vectors of the triplet of data items ($\mathbf{x}_\mathrm{p}$, $\mathbf{x}_\mathrm{a}$, $\mathbf{x}_\mathrm{n}$) are compared using pairwise distance d (L_2 distance) in the Triplet network such that $d(\mathbf{x}_\mathrm{a}, \mathbf{x}_\mathrm{p})$ and $d(\mathbf{x}_\mathrm{a}, \mathbf{x}_\mathrm{n})$.

$$d\left(\mathbf{x}_\mathrm{a}, \mathbf{x}_\mathrm{p}\right) = \sqrt{\sum_{i=1}^{n} \left(\mathbf{x}_{\mathrm{a}i} - \mathbf{x}_{\mathrm{p}_i}\right)^2}$$

$$d\left(\mathbf{x}_\mathrm{a}, \mathbf{x}_\mathrm{n}\right) = \sqrt{\sum_{i=1}^{n} \left(\mathbf{x}_{\mathrm{a}i} - \mathbf{x}_{\mathrm{n}i}\right)^2}$$

Once the distance is computed for $d(\mathbf{x}_\mathrm{a}, \mathbf{x}_\mathrm{p})$ and $d(\mathbf{x}_\mathrm{a}, \mathbf{x}_\mathrm{n})$, we accept the smaller distance to be which image class of $\mathbf{x}_\mathrm{p}$ or $\mathbf{x}_\mathrm{n}$ the anchor $\mathbf{x}_\mathrm{a}$ belongs to. While training, the loss computed for the Triplet network was the Triplet Margin Loss [50] defined as L_T is shown below. The Triplet Margin Loss uses a margin m of 0.55, that was empirically determined prior to evaluation of the approach.

$$L_\mathrm{T}(\mathbf{x}_\mathrm{a}, \mathbf{x}_\mathrm{p}, \mathbf{x}_\mathrm{n}) = \max\left(d\left(\mathbf{x}_\mathrm{a}, \mathbf{x}_\mathrm{p}\right) - d\left(\mathbf{x}_\mathrm{a}, \mathbf{x}_\mathrm{n}\right) + m, 0\right)$$

4.3 Results

During experimentation, the Triplet Classification Networks were run on both datasets of GeneNet as described in Section 3.1. When developing the GeneNet datasets, the performance of the Triplet classification network was considered to ensure the best assortment of publication figures was used. We found in particular that a selection of publication figures for non-models image class performance was over-fitting to only models due to the lack of visual diversity over non-model imagery. The cause of the lack of visual diversity can be attributed to the sampling mechanism not randomly selecting publication figures, but selecting images sequentially. Both GeneNet datasets were separated into a training, validation, and testing split of 70, 10 & 20 respectively.

We present our evaluation results of the proposed Triplet classification network on our GeneNet datasets in Table 4.1. We consider both the best performing accuracy on each dataset and the average of 5 runs to validate the best overall approach. We compare the results of the four feature extraction networks including GeneModelNet, VGG19, AlexNet, and SqueezeNet 1.1. Furthermore, we explore the evaluation performance of VGG19, AlexNet, and SqueezeNet 1.1 when the weights of the feature extraction network are frozen and unfrozen. When unfrozen indicated by "FT", the pre-trained weights are fine-tuned to genetic models using transfer learning [30, 56]. When applying this approach of transfer learning to fine-tune the weights, we use the loss computed by the Triplet classification network similar to our training technique applied to GMN.

Model	GeneNet-98		GeneNet-500	
	Best	Avg	Best	Avg
GMN144	90%	70%	85.15%	80.60%
GMN1028	90%	82%	84.16%	69.90%
VGG19	**95%**	**91%**	93.07%	90.69%
AlexNet	85%	81%	94.06%	**91.49%**
SqueezeNet	75%	26%	1.98%	0.79%
VGG19-FT	90%	82%	**95.05%**	66.93%
AlexNet-FT	85%	83%	91.09%	88.32%
SqueezeNet-FT	35%	7%	2.97%	1.19%

Table 4.1. Testing accuracy for Triplet Classification Networks on both GeneNet datasets.

All of the network conditions were static on all aforementioned approaches when collecting evaluation results. These hyperparameters include a learning rate *lr* of 1×10^{-4} using the Adam optimizer trained for 20 epochs. It was found that due to the small size of the GeneNet datasets, more than 20 epochs would begin to over-fit. The pre-trained networks are loaded from PyTorch's model zoo.

We can conclude that overall the best performing approach of the Triplet classification network was with the use of VGG19 as the feature extraction network. Our hypothesis was confirmed that although we cannot use the image classification layers of a pre-trained feature extraction network, it can still produce viable representations of genetic models.

Chapter 5 Transcription

In this chapter, the two steps required to develop a textual description from a genetic model are described. These steps make up the second and third pillars of computer vision applications for our system to reason about genetic models.

5.1 Region Detection

In this section, we will discuss the second pillar of our pipeline. We are interested in effectively detecting objects classes with genetic models from our GeneNet and GeneNetSyn datasets discussed in Sections 3.1 and 3.2 respectively. As our region detection application, we leverage a pre-trained version of YOLOv5, which is fine-tuned for genetic models using instance-based mapping transfer learning and domain adaptation.

5.1.1 Methodology

We introduce our application of object detection being applied to genetic models which we refer to as region detection throughout this thesis. Our approach to region detection revolves around using learning techniques to transfer the weights from a natural imagery dataset to our publication figures. Specifically, we are interested in transferring the weights from networks trained on natural imagery discussed in Chapter 2.2.1, to our genetic model datasets: GeneNet-500, and GeneNetSyn. Data dependence is a serious problem in deep learning approaches, which alludes to the strong need for large amounts of data to effectively train a network. Insufficient training data is an inescapable issue in certain tasks within computer vision which makes it extremely difficult to solve given problems.

We employ two learning techniques to transfer the weights from Microsoft COCO [4] to our synthetic genetic models within GeneNetSyn. Firstly, we use instance-based mapping transfer learning [30, 56] to tune the weights from the source domain to our target domain. Secondly, to also improve the transition of natural image trained weights to genetic models, we introduce domain adaptation to our region detection networks training approach discussed in Chapter 2.2.4. Although our region detection networks are being trained on GeneNetSyn, we add real genetic models from the GeneNet datasets to make the transition from synthetic to real genetic models have less impact on performance. A visualization of the YOLOv5 training approach can be seen in Figure 5.1

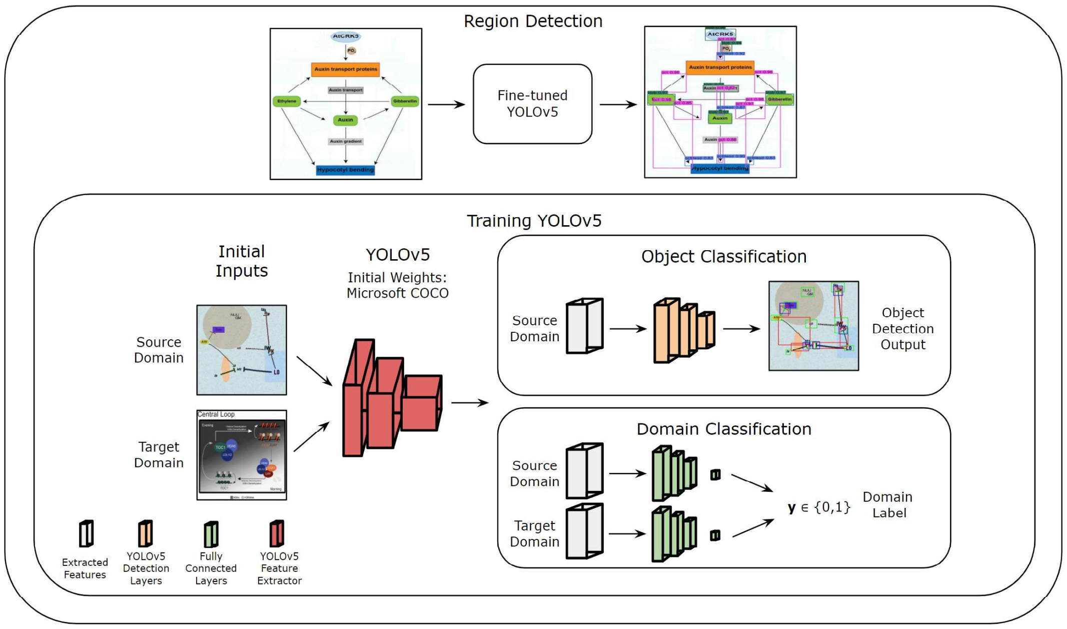


Figure 5.1. Breakdown of the region detection network used for genetic models. An implementation of YOLOv5 and tuning the pre-trained Microsoft COCO [4] using various learning techniques. Genetic model courtesy: Steber *et al.* [18].

One major challenge when attempting to train an application of computer vision for publication figures, more specifically genetic models, is the lack of annotated images. The goal of our second pillar of work is to detect object classes from GeneNet imagery which does not have a large collection of richly annotated imagery. Subsequently, datasets with richly annotated data will need to be synthetically generated as discussed in Section 3.2. During the use of instance-based transfer learning, we translate the 80 object classes from Microsoft COCO [4] to our five object classes in GeneNet and GeneNetSyn discussed in Section 3.2.1.

The object detection network of choice was You Only Look Once (YOLO) version 5 [22]. YOLOv5 is introduced in Section 2.2.2, the specific implementation of YOLOv5 used is discussed in Appendix A.1.2. The selected variation of YOLOv5 used for our region detection network was YOLOv5m. YOLOv5m was found to be the best balance between computational size and performance. Computational size is specifically important as a future goal of the whole proposed system is to be deployed as a rest-API.

5.1.2 Experiments

The experiments conducted to verify the success of effectively detecting biological diagrams using GeneNetSyn will be outlined in this section. Our experiments include a preliminary performance check of the YOLOv5 network trained on a natural image dataset (Microsoft COCO); an iterative training approach to increase the complexity of the dataset in a step-by-step pattern by introducing new datasets with initial sets from the previous dataset; and standard single dataset training.

5.1.3 Base YOLOv5m Performance

We first explore using YOLOv5m pre-trained weights on Microsoft COCO on our test suite of diagrams and show why these weights are not desirable for use. Before applying transfer learning to the YOLOv5m network, we were interested in visualizing the performance of natural image datasets to see what predictions are made. As shown in Figure 5.2, the network trained on Microsoft COCO cannot reason about genetic models. Considering the network was not trained for the relevant classes of interest, these results

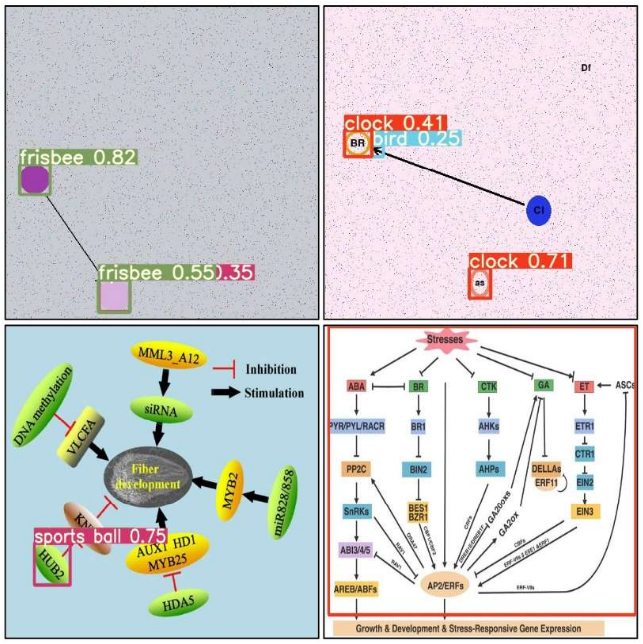


Figure 5.2. Microsoft COCO [4] pre-trained YOLOv5 40% confidence. Genetic models shown courtesy: [1, 19].

are to be expected.

Our experiments on our test suite of example images are evaluated using a 40% confidence to give YOLOv5m a chance to identify anything it possibly can. The network identifies the bounding boxes of some blobs, yet classifies them as some of the following classes: frisbee, clock, sports ball, or any other circular in shape class. Furthermore, relationship arrows and their respective lines are occasionally mistaken for curved objects such as birds.

The performance of YOLOv5m's baseline reflects the conflict of using these extensively trained networks on natural images to not be desirable for visual illustrations. Nonetheless, these pre-trained weights can be valuable for training a network specifically for visual illustrations. Although not trained on our specific domain yet, the value of these pre-trained networks is that they can effectively formulate a solution based on pixel-wise similarities to other known classes. These weights can be used as a starting point for applying transfer learning, rather than using normalized weights. This will encourage faster convergence towards our desired target domain of genetic models trained on GeneNetSyn and to be deployed on GeneNet.

5.1.4 Iterative Training

While creating generations of datasets, it is important to evaluate the performance of YOLOv5m during the transition to a new dataset. In our initial experimentation using iterative training, only three object classes were used for GeneNetSyn. Specifically, the information regarding the head of the relationship was excluded. When training object detection networks it is important to ensure problem areas within the datasets can be isolated and improved. As GeneNetSyn was developed to be more complex, the performance of the trained network needs to be monitored. Thus, when new datasets are introduced we propose training our YOLOv5m network iteratively on the previously trained datasets weights to validate if the network can quickly adapt to changes.

Index	Name	Epoch Count
1	Circular	100
2	Text Only	100
3	Circular Text	100
4	V Blob Circular	100
5	V Shape V Text	100
6	V Curves	50
7	V Dashes	50

Table 5.1. Training steps taken through GeneNetSyn variations (index 1-7) shown in visualization Figure 5.3. V denotes Variable.

To briefly describe the approach for training iteratively, we train from GeneNetSyn dataset index 1 (variable circular) to index 7 (variable dashes) outlined in Section 3.2.1. Each of the first five datasets was trained for 100 epochs, and the two remaining only under 50 to avoid over-fitting. We break down the iterative training pattern for the datasets in Table 5.1. Throughout the aforementioned experiment, the same hyperparameters and learning conditions were used.

As shown in Figure 5.3 when we introduce more complex datasets especially variable curves and variable dashes, we see a downward trend inaccuracy. One of the key

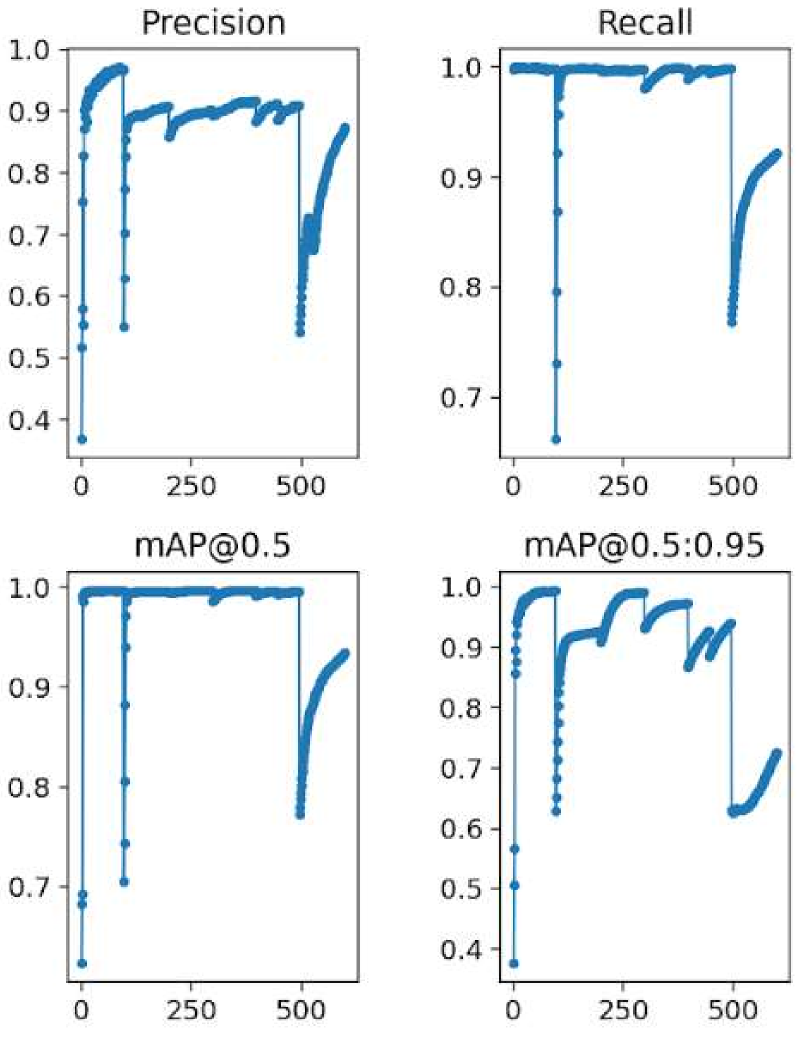


Figure 5.3. Testing results of transfer learning applied to YOLOv5m over the first seven GeneNetSyn datasets (Index 1-7) over 600 epochs.

reasons why this is likely is the introduction of longer textual sequences going from 2 to width dependent (maximum of 6). For 500 epochs, the system has been trained to only detect blobs that were somewhat evenly distributed concerning width and height. The most complex datasets (Indexes 7-9) attempt to challenge the network by not only changing the textual sequence length but inherently larger blobs and relationships.

Although these iterative experiments we conducted verified that the network will begin to struggle as we introduce more complex datasets, it can still perform moderately well on images from GeneNet datasets in Section 3.1. A demonstration of the perfor-

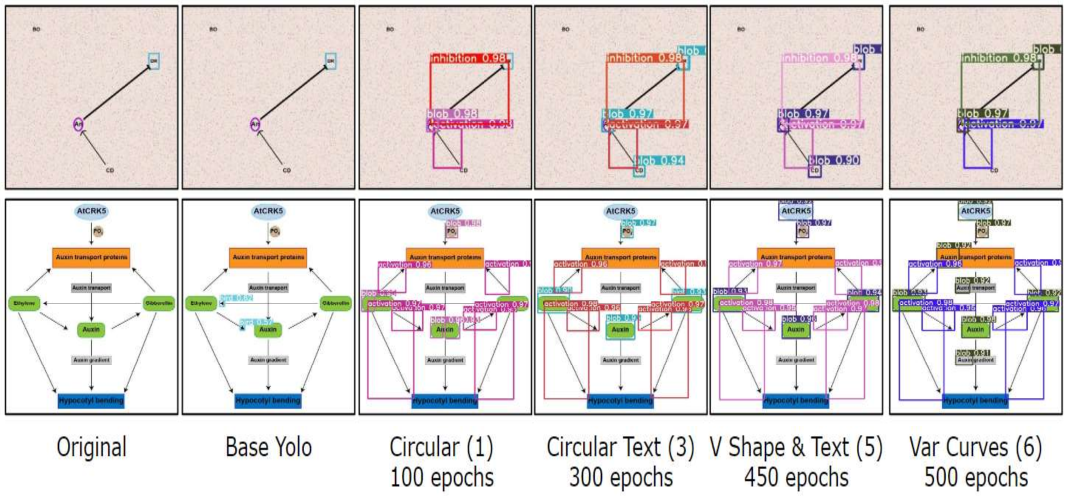


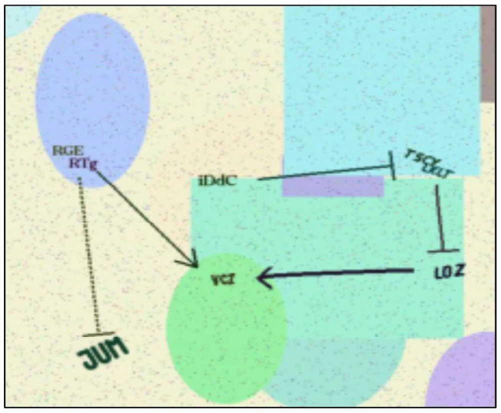
Figure 5.4. Iterative training results throughout the various GeneNetSyn datasets. Particular steps of the experiment were excluded for visualization purposes. Weights used from results shown in Figure 5.3. Dataset indexing used from Section 3.2.1. Genetic model shown courtesy: [57].

mance over the iterative training steps is shown in Figure 5.4.

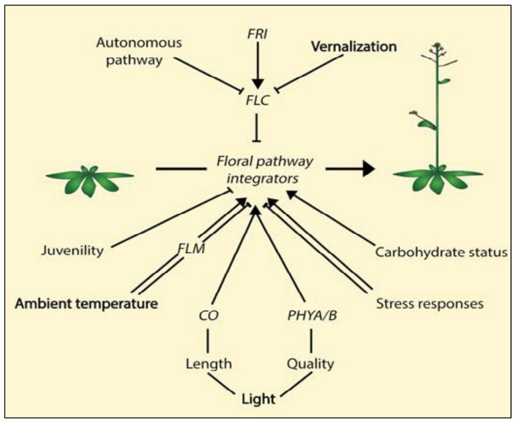
5.1.5 Standard Training

Following the iterative development of datasets and our experimentation during the introduction of more complex datasets, we were interested in finding the best-performing network on genetic models found in GeneNet. Our approach using our standard training methodology includes the previously mentioned instance-based mapping transfer learning on YOLOv5m pre-trained weights from Microsoft COCO. Furthermore, we also propose an addition to include domain adaptation discussed in Section 2.2.4, into the training approach by introducing images from GeneNet for a domain classification. The introduction of domain adaptation helps to improve the transition from our training set of synthetic diagrams to real biological diagrams considered the domain shift. A discriminative classifier to predict the domain is added to help tune the weights trained on our source domain (GeneNetSyn) to improve the evaluation performance on the target domain (GeneNet).

Our standard training experiments were conducted considering all five object classes of our GeneNetSyn dataset outlined in Chapter 3.2. When finding the most optimally



a: Synthetic Genetic Model



b: Real Genetic Model

Figure 5.5. Comparison between the source domain of GeneNetSyn (a), to the target domain of GeneNet (b). Right image courtesy: Bloomer *et al.* [8].

performing network it must gauge good performance while training on synthetic diagrams, but its ability to translate over to imagery from GeneNet is essential. Comparing each dataset and the two approaches including the default training mechanism and domain adaptation, results were demonstrated in the quantitative evaluation of our diagram parsing approach in Chapter 5.2.3. For the aforementioned experiment, we consider a case-by-case evaluation with the resulting output metrics by Ultralytics YOLOv5 which collects per epoch: precision, recall, mAP@0.5, and mAP@0.5:0.95. Following initial tests of the earlier datasets of Version 1 in GeneNetSyn which showed their trained networks are not suitable for the next pillar of work. Thus, the datasets from GeneNetSyn compared include the following: Variable Dashes, Variable False Lines, Version 2, and Version 3.

The second pillar of region detection for our proposed system is considered the most essential to accurately determine entity-relationship-entity triple sets for the third pillar. Without effectively detected regions of interest, our later steps subsequently fail as it does not have the required information to parse the genetic model. The general hypothesis is that the most complicated synthetic genetic model dataset (GeneNetSyn Version 3) would perform best on GeneNet imagery.

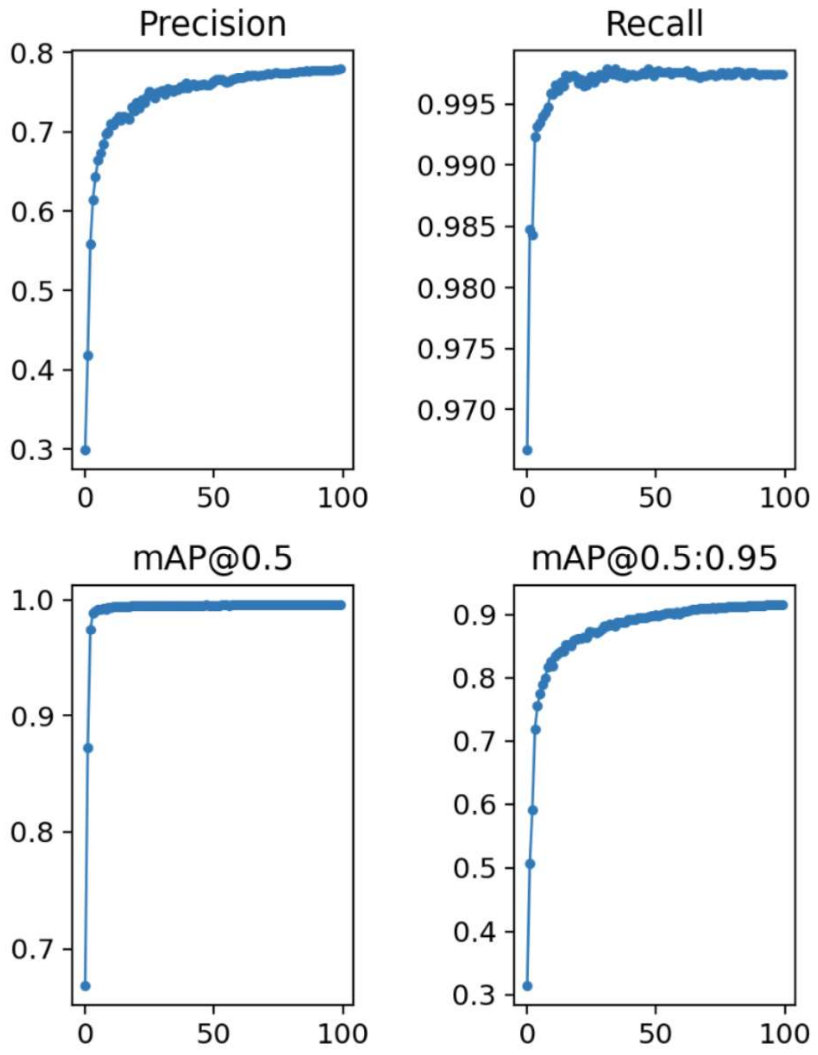


Figure 5.6. Testing results of YOLOv5m over 100 epochs on GeneNetSyn Version 3 dataset.

The performance of YOLOv5m over 100 epochs on the aforementioned datasets from GeneNetSyn can gauge how well trained the network is shown in Figure 5.6. To visualize the performance metrics, we consider our most complex dataset being GeneNet-Syn Version 3 that collects strong evaluation results on both the mAP@0.5:0.95 and precision metrics. For the sake of discussion, the result metrics of previously less complex datasets are comparable yet perform worse on imagery from GeneNet. Based on the limitation of a richly annotated set of real biological diagrams, hypothetically the synthetic diagram datasets should roughly represent the performance of real biological diagrams.

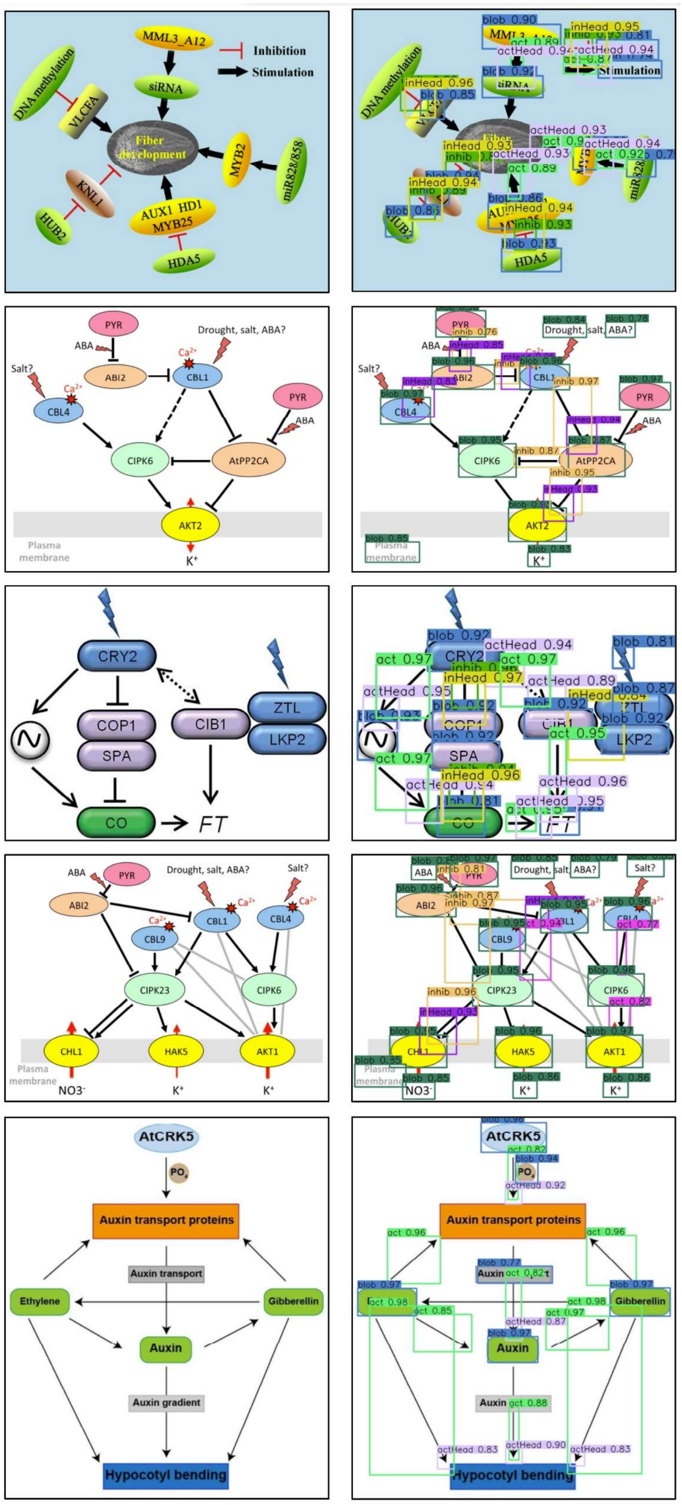


Figure 5.7. Best performance of YOLOv5m trained on 100 epochs of GeneNetSyn Version 3 dataset. YOLOv5m's detection confidence set to 75%. These images were ran at their base resolution. Genetic models shown courtesy: [1, 57, 21, 16].

We conclude that training a network on only our most complex dataset is desirable when shifting domains to our target domain of real genetic models. Compared to our performance as we iterate over datasets shown in Section 5.1.4. To ensure that our synthetic dataset is a viable training technique for the evaluation of real biological diagrams, we must inspect the performance qualitatively over all experiments on a test suite of images from the GeneNet dataset.

We present various examples of our synthetic image trained YOLOv5m applied to GeneNet images shown in Figure 5.7, which gives us a good idea of how much we have advanced the application of object detection to publication figures and genetic models. The GeneNetSyn Version 3 trained YOLOv5m can identify upwards of 100% constituents present within a synthetic diagram with the confidence of over 90% on most of our test suite of images.

Throughout the presented experiments, it was found that detection confidence of 75% on real genetic models produced the best detection performance. This detection confidence represents the minimum accepted confidence of a detected object class, thus an object detected with less than 75% would not be accepted. This confidence currently offers the best balance between finding the most constituents in the diagram and making the least amount of mistakes. We find that the network can effectively find a large number of textual elements classified as blobs with a high degree of confidence.

As demonstrated in our numerous examples of genetic models from GeneNet using YOLOv5m, our system can locate a majority of regions of interest. Unfortunately, due to the limitations of richly annotated real genetic models, we cannot provide upwards of human-level accuracy when shifting domains from synthetic to real diagrams. Below, we will discuss a few of the shortcomings of our approach to detecting object classes on real genetic models from GeneNet which hinders its performance:

1. **Large blobs**: Isolating one bounding box around a large blob. Some ways a large blob can be formed include extremely large font; large blobs may contain multiple

textual features in very close proximity (association); and textual features which have special characters separating a sequence of text.

2. **Long relationships**: Relationships that span across the whole image, not well represented in GeneNetSyn due to the random nature of blob and relationship placement. Long relationships can also represent large relationships in real genetic models which are substantially bigger in dimensions, thus relationship/blob scaling being much larger.
3. **Underlined text**: Our system has yet to be trained to understand blobs in which whose textual features have an underline, which causes it to occasionally classify an underline paired with a character to be a relationship.
4. **Text above relationships**: Although considered two different object classes, our system and datasets were not developed to understand a middle textual feature above a relationship. Often being smaller fonts tight to the relationship line, our system often mistakes text above relationships as part of the relationship visual characteristic.

5.2 Diagram Parsing

In this section, we discuss the third pillar of our proposed system for diagram parsing on biology publication figures. We introduce our algorithm that from the regions detected and textual information of an image, entity-relationship-entity triplet pairs are determined. These entity-relationship-entity triplet pairs are considered the final output of our proposed system. Later, we present both the qualitative and quantitative results of our approach.

5.2.1 Methodology

The goal of the third pillar of our diagram parsing system is the ability to parse relationships from the object classes detected from pillar two discussed in Section 5.1. Parsing relationships within diagrams was initially proposed in 2016 using an LSTM based approach by Kembhavi *et al.* [2] discussed in Chapter 2.1, thus motivating our approach. One important consideration we took into account when developing our approach is the hardware requirement if this system were to ever be deployed as a rest-API. The development of our approach on a rest-API for commercial use is considered future work. We require a diagram parsing system without a large computational requirement.

Algorithm 1 Diagram parsing approach for a given genetic model (GMM)

```
Require: $D$: GMM Data Item
  $img \leftarrow D[0]$: GMM image
  $d \leftarrow D[1]$: GMM region detection output
  $t \leftarrow D[2]$: GMM text features
  $R \leftarrow \emptyset$: Container for triplet sets
  $BL \leftarrow \emptyset$: Best line from active contour model
  Collect relationship b-boxes $B \subset d$
  Collect relationship head b-boxes $H \subset d$
  Collect blob b-boxes $F \subset d$
  for all $b_{\mathrm{i}} \in B$ do
      $LR, RL \leftarrow ACM(img, p)$: Get active contour model with hyperparameters ($p$)
      $PLR, PRL \leftarrow PF(LR, RL)$: Compute polyfit of $LR$ & $RL$
      if $\sqrt[2]{\|LR\| + \|PLR\|} > \sqrt[2]{\|RL\| + \|PRL\|}$ then
          $BL \leftarrow LR$
      else
          $BL \leftarrow RL$
      end if
      $CH \leftarrow \min \forall h_j \in H(\|h_0 - b_{\mathrm{i}}\|, \cdots, \|h_{\mathrm{j}} - b_{\mathrm{i}}\|)$: Closest relationship head to $b_{\mathrm{i}}$
      $TB \leftarrow \min \forall f_j \in F(\|f_0 - BL_0\|, \cdots, \|f_{\mathrm{j}} - BL_0\|)$: Closest blob to tip of $BL$
      $FB \leftarrow \min \forall f_j \in F(\|f_0 - BL_{\mathrm{k}}\|, \cdots, \|f_{\mathrm{j}} - BL_{\mathrm{k}}\|)$: Closest blob to tail of $BL$
      $ct \subset t$: Extract text features for $TB$ & $FB$
      $R \overset{+}{\leftarrow} (TB, b_{\mathrm{i}}, CH, FB, ct)$
  end for
```

We introduce our approach to parsing entities associated with a relationship. We leverage the use of an active contour model [7] which was described in Section 2.2.5. Furthermore, this approach does not use any machine learning inference or prediction. The goal of our task is to generate sets of triplets representing entity-relationship-entities found within a given diagram. The following methodology offers a solution for both GeneNet and GeneNetSyn diagrams. One minor difference between the implementations of the two datasets is how the text within textual features is collected. For the GeneNetSyn datasets, we assume that text to bounding box association are known. When considering

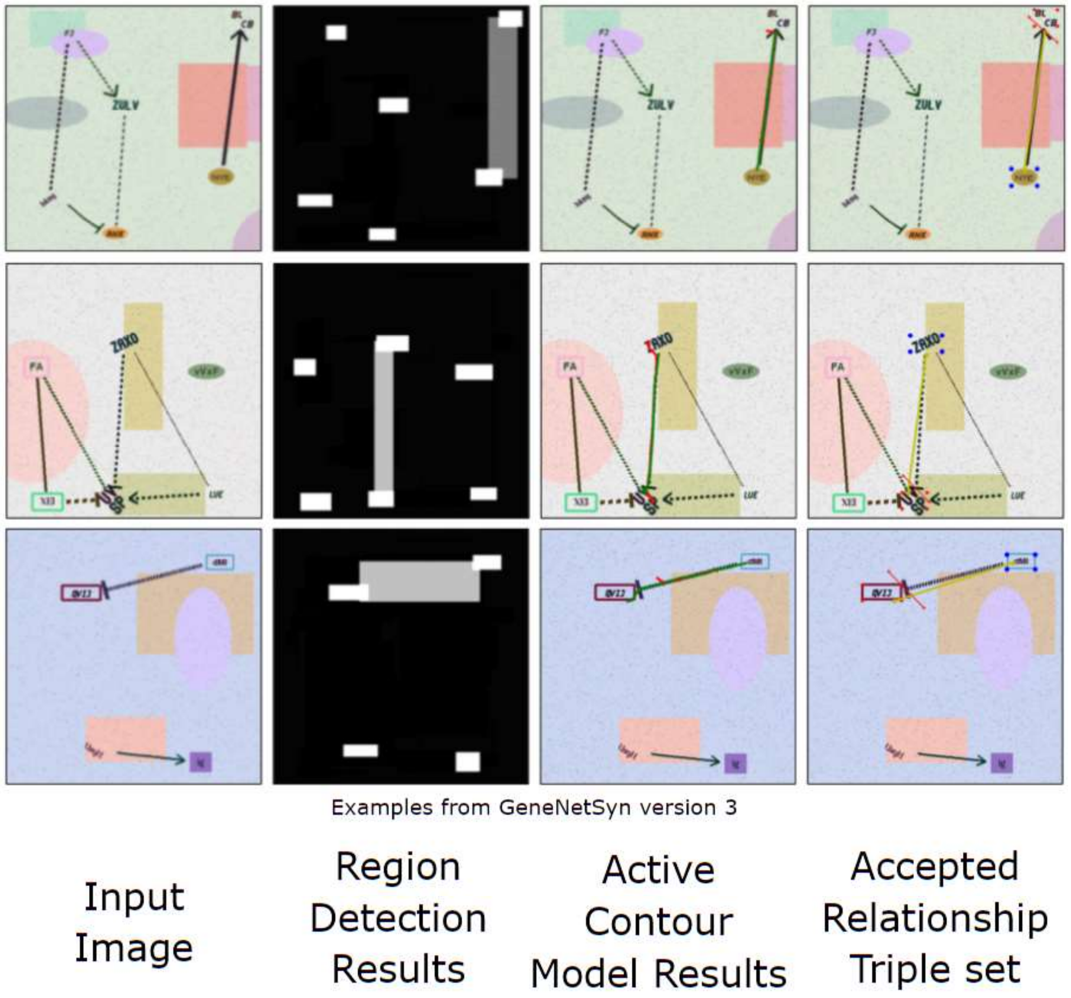

Examples from GeneNetSyn version 3
Input Image
Region Detection Results
Active Contour Model Results
Accepted Relationship Triple set

Figure 5.8. Full diagram parsing parsing algorithm summarization broken into four components. Example images used from GeneNetSyn Version 3.

diagrams from GeneNet or other genetic models not known, we would have textual information located using optical character recognition discussed in Appendix A.3. We can divide our approach into three steps which we will specifically identify and describe: initialization; relationship pairing; and entity-relationship-entity. We outline the proposed solution for our diagram parsing approach in Algorithm 1, which is supplemented Figure 5.8.

Initialization

The initialization step of our approach includes the collection and organization of information collected in prior steps of the full pipeline discussed throughout this thesis. Assumptions to this approach is that the algorithm includes the following: a genetic model; textual feature information; and the class and bounding box of each detected region. Given a diagram of interest deemed a genetic model, we initially begin with: the RGB image, the bounding boxes from our region detection system, and the textual contents of the image. An example of the region detection output which would be our initial data item can be seen in Figure 5.9.

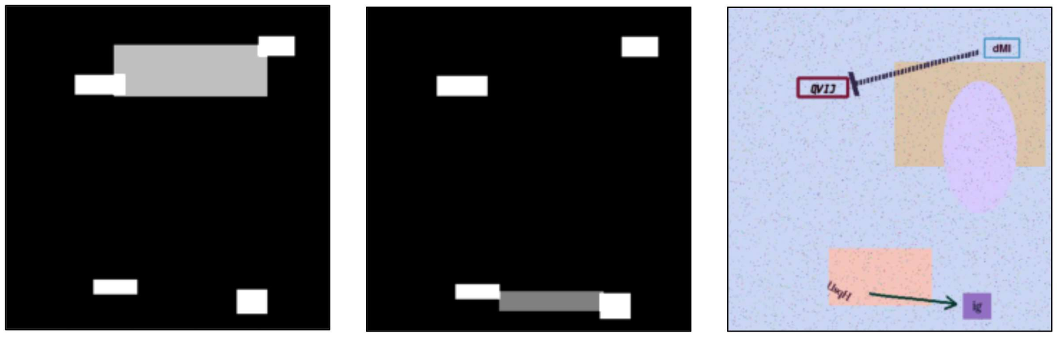


Figure 5.9. Initial data of our diagram parsing approach. Demonstrating the given bounding box information using gray-scale to represent locations from original image (right). Inhibition relationships shown in light gray (left image), activation relationships shown in dark grey (middle image), and blobs are represented by white regions.

Relationship Pairing

When dealing with a given relationship within an image, the problem of finding the entity-relationship-entity triplet is two-fold. Firstly, the relationship needs to be paired with its respective detected relationship head. The pairing is essential to determine the direction of the relationship such that the entity that the relationship is traveling to can be assumed. Secondly, given the bounding box of the relationship, the line contained within this space needs to be isolated. Isolating the tip and tail of the line present within the relationship is used to determine distances to both the head of the relationship and connected entities.

The goal is to collect the endpoints of each relationship and pair them with their associated relationship head. For each given relationship bounding box, our approach follows the algorithmic steps listed below:

1. Determine the left-to-right (LR) and right-to-left (RL) snakes using an active contour model discussed in Section 2.2.5. The purpose of collecting these snakes is to find where within the given bounding box the line occurs. To instantiate diagonal snakes, we collect the four corners of the bounding box and generate a linear space from the LR (top-left to the bottom-right) and RL (top-right to the bottom-left). Before the image is processed by the active contour model, it is set to gray-scale to encourage edge and line discovery based on given constraints. The hyperparameters of the active contour model were empirically determined through initial experimentation. These hyperparameters were tuned to be optimal for both synthetic and real genetic models. The description of the hyperparameters are as follows:
    - **Boundary condition**: Limitations of how the snake positions can travel. Boundary condition in our implementation was set to *free*.
    - **Alpha**: Snakes shape parameter, a lower value set to make snake contract slower to avoid clumping. Alpha in our implementation was set to 0.0001.
    - **Beta**: Snakes smoothness parameter, a higher value set to smooth the snake through the contraction. Beta in our implementation was set to 2.
    - **W-line**: Snakes attractiveness to brightness, was set to negative to attract the snake towards dark pixels. W-line in our implementation was set to 5.
    - **W-edge**: Snakes attractiveness to edges, set to attract towards edges. W-edge in our implementation was set to 1.
    - **Gamma**: Explicit step time. Gamma in our implementation was set to 0.0005.
2. The resulting output for both LR and RL snake contours is shown in Figure 5.10. Following the active contour model output, it is clipped around the bounding box to ensure the tip and tail of the snake are not outside. Snake points outside of the

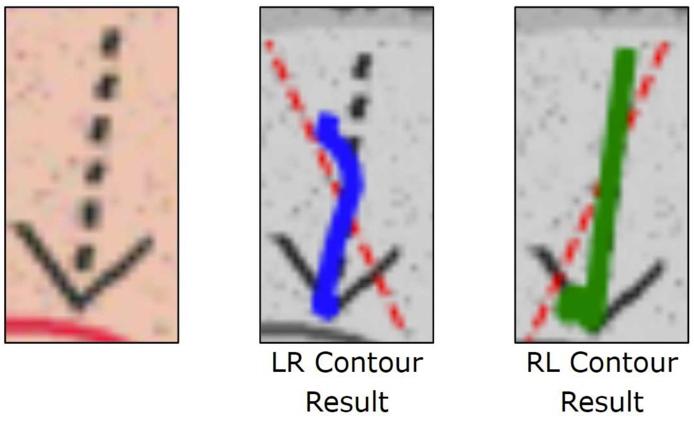


Figure 5.10. Given the relationship bounding box (left), the LR (middle) and RL (right) contours are computed. The snake is initialized represented by the red dashed diagonal line. The resulting snakes are representing in blue (LR) and green (RL).

bounding box are not desirable when collecting the endpoints because due to the nature of biological diagrams, often other relationships and non-related entities are nearby. Once any snake points outside of the current bounding box are removed, we accept one of the outputs as the predicted *valid* line. In most cases, both snakes follow a similar direction due to attraction based on dark regions (lines and edges) present within the bounding box. When selecting the best fit snake, the endpoints of both snakes are passed also to a polynomial curve fitting model to flatten the snake. We then take the output of both the ploy-fitted snakes and original snakes and take the average between both variations of the LR and RL snakes. The best line is chosen based on the average of the original and poly-fitted output for the LR and RL snake. This average is then compared to the magnuitude of the relationship bounding box, where the smaller difference is chosen as the best line.

3. Once the tip and tail of the best line are collected, we lastly need to determine the head associated with the given line to complete our parsing of the relationship. Selecting the head of the relationship is a distance-based metric concerning the center of the relationship head's bounding box to either the tip or tail of the best line. We select the relationship head which the center of its bounding box has the smallest distance to either the tip or tail of the line of interest. We can assume

that the head we desire should be near the accepted best line of the active contour model. It is assumed that the relationship head will occur on one of the corners of the relationship's bounding box, yet we found greater accuracy using the points of the best line due to relationship clutter.

Entity-Relationship-Entity Selection

Following the relationship paring step of our algorithm, we have both the active contour model's best-associated line and the accepted relationship head for all relationships present in the diagram. With this information, we can now create entity-relationship-entity triplet sets that can be used to describe textually what interactions are present in the diagram. Creating our triplet sets requires that the relationship head is correctly detected within the relationship, as the head information is used to determine the inherent direction of the relationship. The direction will indicate which entity is at the beginning (from-blob) and which entity is at the end of the relationship (to-blob). Parsed triplet sets will output a textual sentence of the form: "HSTE activates YBSZU".

To collect our two required entities for the current relationship, the closest detected blob to both the tip and tail needs to be found. A demonstration of a parsed synthetic genetic model through this approach can be seen in Figure 5.11. We iterate through the detected blobs and measure their center point to both the tip and tail locations, storing the smallest distance which is initially set to a default value equal to the height and width of the image. Distances are determined using a 2-norm from each of the endpoints of the best line. Once the shortest distances are determined, we accept this as our final output for the given relationship. Furthermore, before the parsing step, the textual entity which is already known is paired with the respective blob bounding boxes selected to be the from-blob and to-blob. Finally, we take the textual entities of the to-blob and from-blob and connect it with the found relationship to create (from-blob text, relationship type, to-blob text) which is presented to the user.

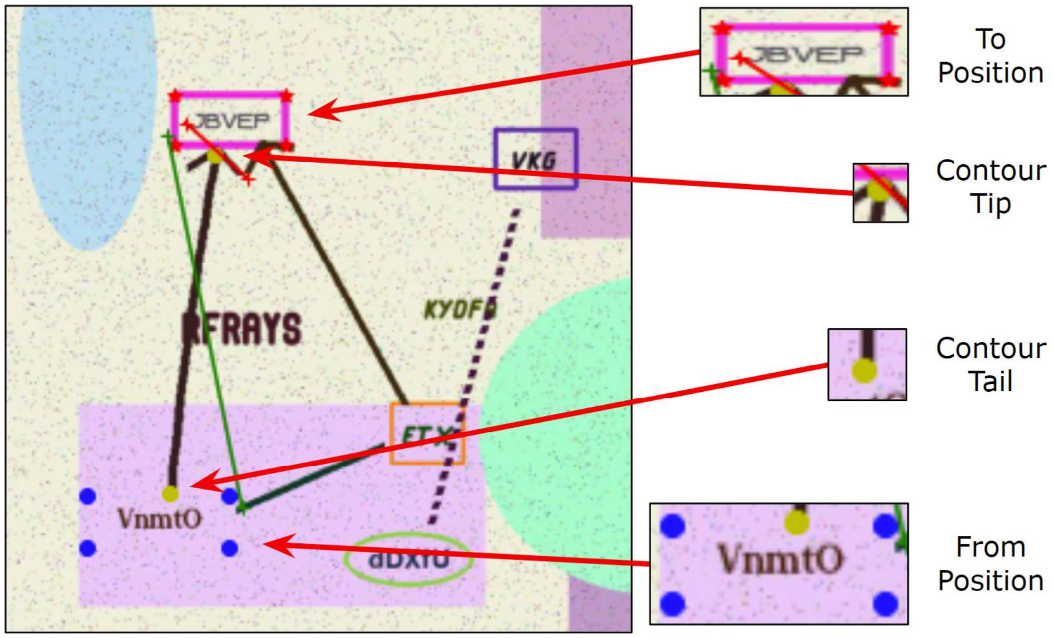


Figure 5.11. Breakdown of each feature of the entity-relationship-entity triplet sets parsing. The accepted relationship snake contour represented by the green line, and the relationship head by the red line. Resulting relationship: VnmtO activates JBVEP.

5.2.2 Discussion

In our discussion, we consider a collection of examples of our aforementioned relationship parsing approach and evaluate each qualitatively. To describe the success and shortcomings of the aforementioned approach we use visual examples from both GeneNetSyn and GeneNet datasets.

Demonstration 1

First we consider a genetic model from GeneNetSyn Version 3 shown in Figure 5.12. This diagram contains six blobs, three inhibition relationships, two activation relationships, and each relationship's associated head. Our goal to achieve 100% accuracy on this diagram is to produce the following output: RGE RTg inhibits JUM; RGE RTg activates VCZ; iDdC inhibits TSCY CXLT; TSCY CXLT inhibits LOZ; and LOZ activates VCI. Initially, our algorithm begins with the region detection output which successfully extracts all regions of interest, which are represented by a gray-scale visualization demon-

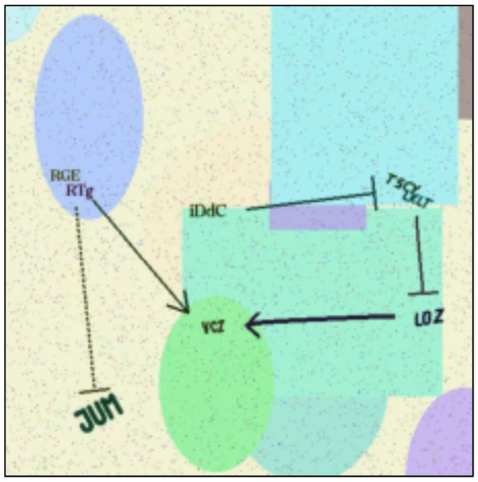


Figure 5.12. Image #11005 from GeneNetSyn Version 3.

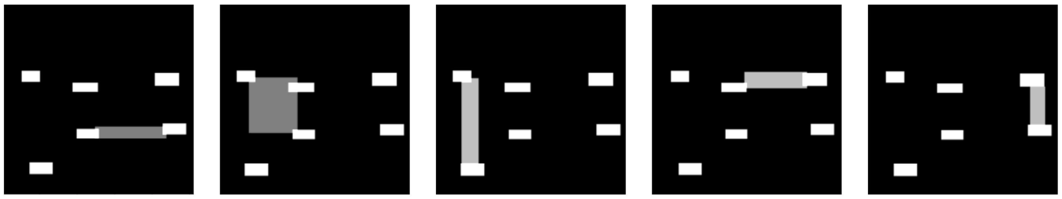


Figure 5.13. Region detection output visualized in gray-scale, white regions indicating blobs and gray regions being relationships.

strating both the class and location of the bounding boxes. As shown in Figure 5.13, we isolate each relationship and consider each individually with respect to all of the blobs to determine our entity-relationship-entity triplet set.

Following the steps outlined in the methodology, we are left with enough information to generate our triplet set. This information includes the best line determined by the active contour model, the relationship head bounding box, the bounding boxes, and the textual features of the two blobs which reside closest to the tip and tail of the best line as determined by the active contour model result. We visualize these results in Figure 5.14.

The algorithm correctly identifies each of the entity-relationship-entity triplet sets. Furthermore, we plot the tip and tail of the active contour model snake, which is seen as the yellow line connecting two points (tip and tail). If we consider the left-most and right-most sub-images in Figure 5.14, we can see that both the tip and tail (yellow dot) reside

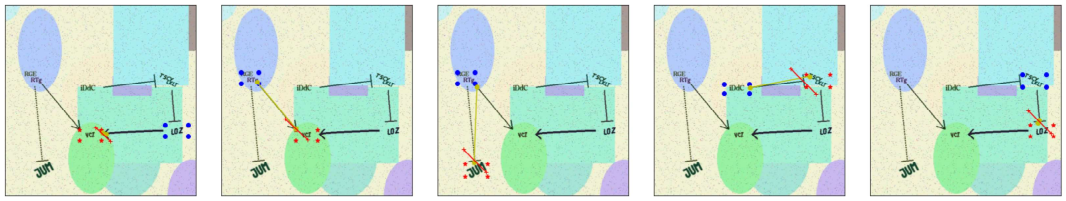


Figure 5.14. Parsed output from our diagram parsing algorithm. The to-blob and from-blob are identified with blue and red markers respectfully. The tip and tail of the active contour model are connected with a straight line in yellow, and the relationship head's bounding box being the red line.

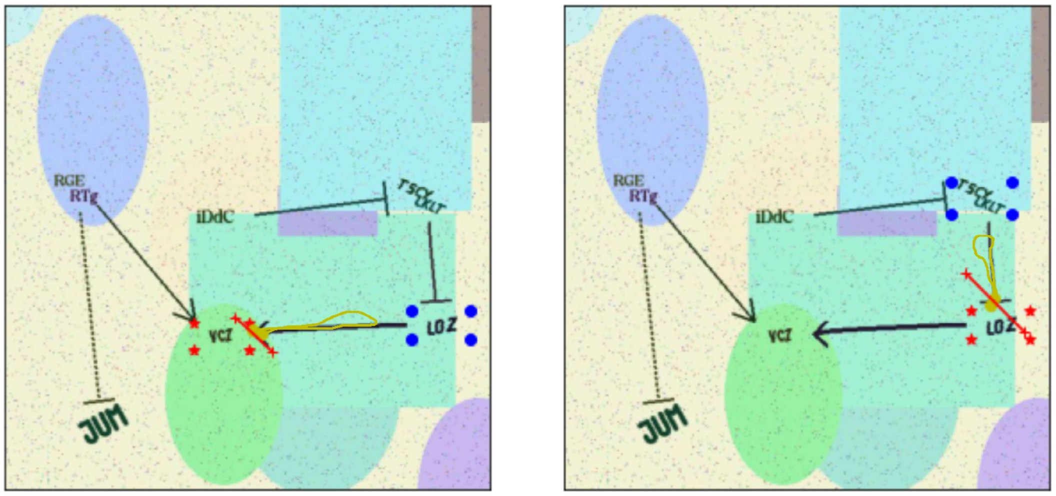


Figure 5.15. Demonstration of the potential snake that might be present when both the tip and tail of the best line occur on the same end of the relationship.

near the relationship head. This is one issue present when our active contour model attracts a majority of the snake towards the corners or edges present in the relationship. The reason why the algorithm still effectively determined the correct related blobs is the use of the poly-fitted snake, which helps to straighten the resulting active contour model snakes across the relationship curve/line. In the shown instance, the raw snake output would likely contain a loop from the tip to the tail which also traveled along the relationship line before doubling back to the corner and edges towards the relationship head. This instance is dealt with by a special case that takes the furthest point from the tip and tail of the snake and poly-fits the snake between the tip and the midpoint. Examples of the resulting raw snake that loops are shown in Figure 5.15.

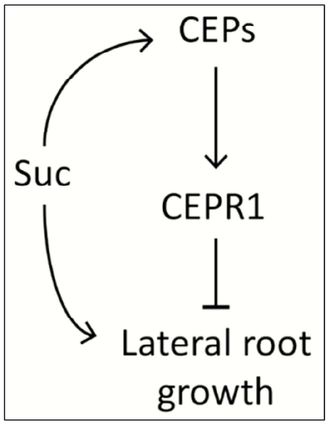


Figure 5.16. Image #75 from both GeneNet-500 and GeneNet-DP. Genetic model shown courtesy: Chapman *et al.* [20].

Demonstration 2

Second, we consider a genetic model from GeneNet-DP shown in Figure 5.16. This diagram contains four blobs, two inhibition relationships, two activation relationships, and each relationship's associated head. Furthermore, the locations of textual features were previously extracted by Google Vision API OCR outlined in Appendix A.3. Although a simple black and white image, this image offers two primary challenges including the long textual feature of "lateral root growth" and long curved relationships. When passed through the region detection network, we are presented with only three relationships shown in Figure 5.17. At initialization of our diagram parsing approach, we are only capable of 75% accuracy on the complete diagram as we are missing the relationship between "suc" and "lateral root growth". One interesting observation is that the blob detected for "lateral root growth" detects two textual features creating a bounding box around "lateral" and "root". Although trained for multiple textual features, our region detection system struggles to isolate a collection of text which is both long and large ("lateral root growth", containing a 375 width and 215 height) considering a 512 width and height dimension training set within GeneNetSyn.

Considering the shortcomings of the region detection network on the given diagram discussed above, we can assume that these issues will cause problems when pars-

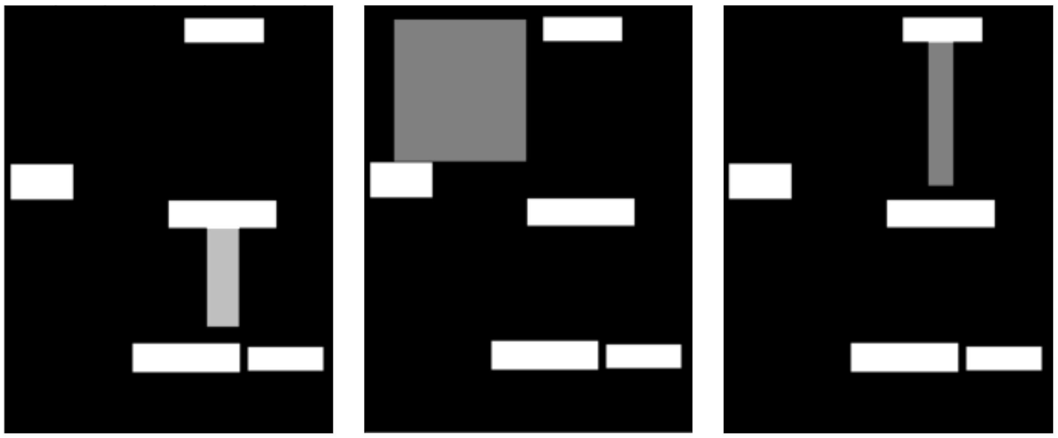


Figure 5.17. Region detection output visualized in gray-scale, white regions indicating blobs and gray regions being relationships.

ing the entity-relationship-entity sets. Although the region detection failed at detecting all relationships present, the relationships which were detected all correctly isolated the entity-relationship-entity set. The resulting diagram parsing output can be seen in Figure 5.18.

This demonstration identifies that our approach of using the active contour model is successful when a given diagram has little background clutter. As shown in Figure 5.18, our tip and tail for each relationship do a suitable job to span the relationship line to allow the distance measurements to effectively select the associated blob (i.e, entities). One mistake which is made by the algorithm is the selection of the “lateral” blob instead of “lateral root growth”. This issue was caused by the region detection network’s failure to isolate the collection of textual features together. For quantitative evaluation, the relationship triplet set of “CEPR1 inhibits lateral” would be marked as incorrect with the correct relationship being “CEPR1 inhibits lateral root growth”. One thing to consider when looking at the quantitative evaluation metrics is that the relationship must select both full and correct blobs, as well as predict its direction properly to count as a correct prediction. We show within this demonstration that although the diagram parsing algorithm can effectively isolate the current relationship, the region detection may cause a limitation of effectiveness.

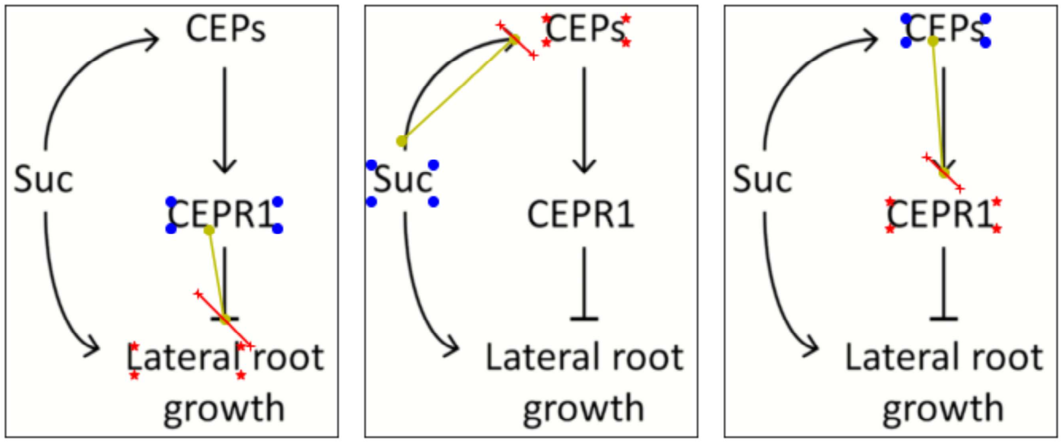


Figure 5.18. Visualization of the result from our diagram parsing approach. The to-blob and from-blob are identified with blue and red markers respectfully. The tip and tail of the active contour model are connected with a straight line in yellow, and the relationship head's bounding box being the red line. Genetic model shown courtesy: Chapman *et al.* [20].

Demonstration 3

Lastly, we consider a difficult genetic model from GeneNet-DP shown in Figure 5.19. This diagram contains nine blobs, six inhibition relationships, four activation relationships, and each relationship's associated head. Furthermore, the location of textual features was previously extracted by Google Vision API OCR outlined in Appendix A.3. This genetic model is challenging for our diagram parsing approach. Genetic models such as this can offer various obstacles which cannot only throw off the region detection network but inherently attract the active contour model snake towards other features in the diagram not associated with the relationship of interest.

We can first consider the initial information extracted by our region detection network in Figure 5.20. One positive note for the region detection network is that all of the required blobs are correctly detected. When considering the relationships detected, only five of the ten relationships present in the diagram are detected which can be due to the size of the diagram or other background clutter close to the relationship. Our concerns for this diagram's parsing through our diagram parsing algorithm are two-fold.

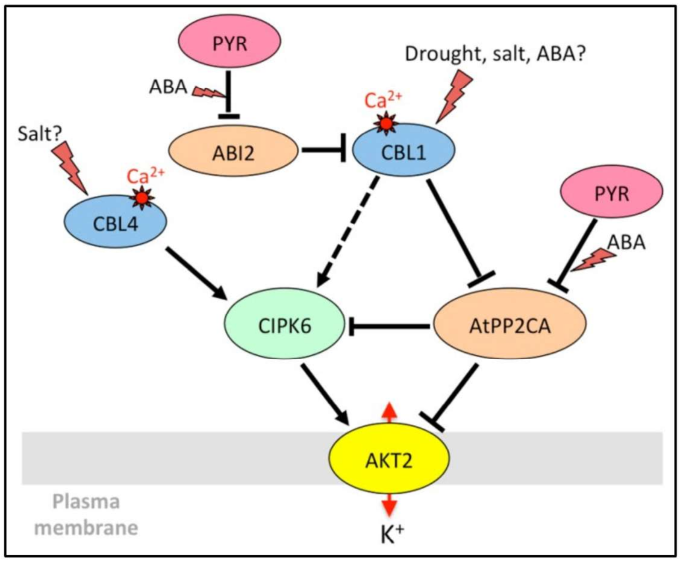


Figure 5.19. Image #130 from both GeneNet-500 and GeneNet-DP. Genetic model shown courtesy: Chérel *et al.* [21].

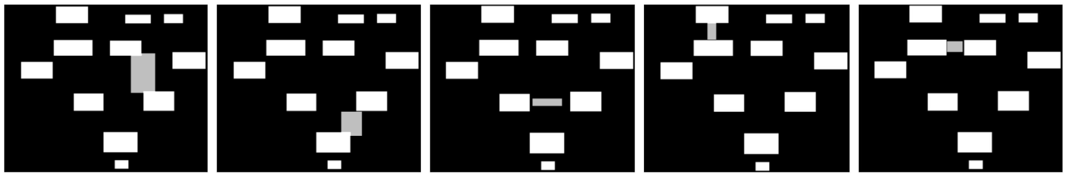


Figure 5.20. Region detection output visualized in gray-scale, white regions indicating blobs and gray regions being relationships.

Firstly, although detecting all of the blobs correctly with a high degree of precision for each bounding box at the top of the diagram both “Drought” and “ABA?” are also included as a detected blob. These textual features not necessarily associated with a relationship such as “Drought”, “salt”, “ABA?”, “ Ca2+”, “Salt?”, and “ABA” which occur all around the genetic model. Although containing valuable information, these blobs can be considered as background clutter or false positives for our diagram parsing approach.

Secondly, other relationship types are present in the diagram that are not known to the region detection network shown as “ lightning bolt” style constituents. These constituents also offer additional edges and lines which the active contour model’s snake typically moves towards, introducing a degree of error for a given relationship’s LR and

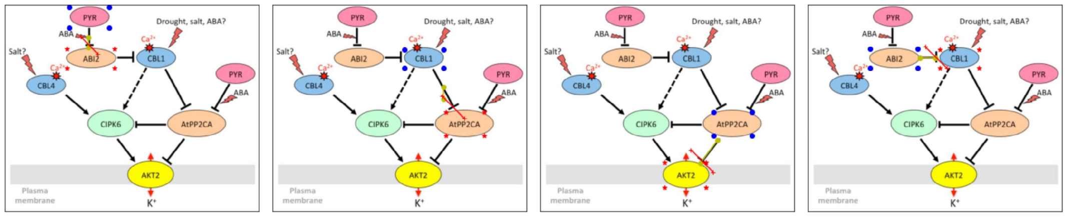


Figure 5.21. diagram parsing algorithm output for relationships which performed well. The to-blob and from-blob are identified with blue and red markers respectfully. The tip and tail of the active contour model are connected with a straight line in yellow, and the relationship head's bounding box being the red line. Genetic model shown courtesy: Chérel *et al.* [21].

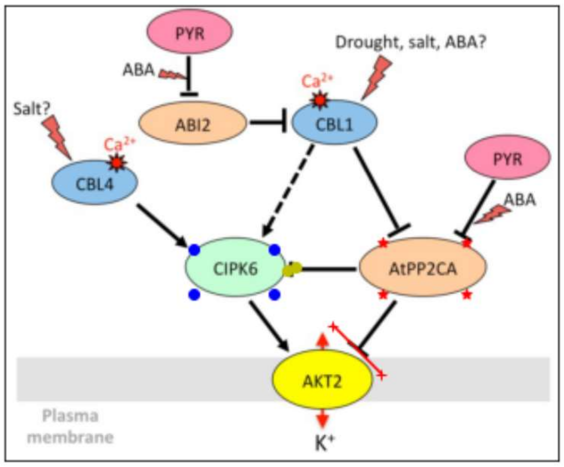


Figure 5.22. diagram parsing algorithm output for relationships with large errors. The to-blob and from-blob are identified with blue and red markers respectfully. The tip and tail of the active contour model are connected with a straight line in yellow, and the relationship head's bounding box being the red line. Genetic model shown courtesy: Chérel *et al.* [21].

RL snake. Additionally, these constituents can throw off the region detection network's ability to effectively classify each relationship.

When considering the diagram parsing algorithm and its resulting output shown in Figures 5.21 & 5.22, we can see 80% of the relationships parsed evaluate to the correct entity-relationship-entity triplet set. Also shown in demonstration 2 with the easier diagram from GeneNet-DP, our limitation of the diagram parsing algorithm is based on the data being passed to it.

Specifically, we isolate the relationship between "AtPP2CA" and "AKT2" shown in Figure 5.22 where one issue with the active contour model is identified. When a collection of relationships form a circular pattern nearby, the active contour model can easily gravitate towards other relationships. When initialized, the active contour model snake will move from its original diagonal across the relationship bounding box towards lines and edges nearby. Thus if other relationship tip and tails are very close to the bounding box of a relationship of interest, there is a high probability that the active contour model snake will move towards a relationship line that is not associated. For example, at initialization, the active contour model snake spans RL and LR of the bounding box the relationship between “AtPP2CA” and “AKT2”. In this instance, the snake is attracted to the tail of the relationship between “AtPP2CA” and “CIPK6”.

5.2.3 Results

We present our results for our diagram parsing approach. Specifically, we are interested in evaluating the ability of our approach to isolate triplet sets (entity-relationship-entity) correctly. Before reaching this stage in our approach, we rely on the high-quality output of the region detection pillar discussed in Section 5.1 respectfully. Our method defined in Section 5.2.1 outlines specifically the algorithm used to generate our triplet sets, the accuracy presented below represents the number of correct triplet set predictions. For a relationship to be included in our performance tables below, the relationship must be detected in the previous step of our system. To ensure a fair evaluation of each method while training YOLO, we provide each variation of the datasets with the same hyperparameters and training length. Furthermore, for our networks trained using domain adaptation (DA) we also show the same set of biological diagrams throughout each training epoch. YOLO networks that did not contain the use of domain adaptation are identified as default. We first consider a comparison evaluating various datasets from GeneNetSyn and GeneNet shown in Table 5.2.

Based on the table 5.2, it is clear that our approach struggles with real genetic models from GeneNet-DP. In contrast, we see the great success of diagram parsing on

Test \ Train	Dashes		False Lines		GNS-V2		GNS-V3	
	Default	DA	Default	DA	Default	DA	Default	DA
Dashes	85.30%	85.28%	86.87%	**86.67%**	81.91%	82.70%	80.89%	82.72%
False Lines	79.99%	79.10%	83.70%	**84.20%**	80.54%	80.00%	78.91%	79.47%
GNS-V2	83.85%	81.54%	90.35%	**90.56%**	88.73%	88.50%	89.05%	88.64%
GNS-V3	50.99%	49.61%	70.16%	70.93%	56.49%	60.78%	88.29%	**88.95%**
GeneNet-DP	39.13%	39.19%	47.22%	49.30%	34.52%	33.33%	38.10%	**45.75%**

Table 5.2. 100 epochs training YOLOv5 on all training sets.

various of the GeneNetSyn datasets. Specifically, we achieved upwards of 89% accuracy for diagram parsing on GeneNetSyn Version 3 being trained with its respective training set also applying domain adaptation.

We describe some of the shortcomings when parsing real biological diagrams in our qualitative evaluation discussed in Section 5.2.2. To extend our results, we manually identified from our set of GeneNet-DP diagrams a collection of both easy and difficult diagrams. We classify easy diagrams to be genetic models which do not contain large amounts of visual background clutter and complicated relationship lines. Thus, our difficult set of diagrams from GeneNet-DP would be considered challenging diagrams that are difficult to replicate, and often contain a large amount of visual characteristic variance.

To further motivate the evaluation performance differences among various diagrams from GeneNet-DP, we offer an additional collection of results shown in Table 5.3. These experiments were using a region detection system trained on GeneNetSyn Version 3 paired with domain adaptation, which we found to be our best overall performing network. We find as expected our approach lacks effective performance on hard diagrams from GeneNet-DP. Our challenge arises from the lack of relationships detected by our region detection network, as our probabilistic approach to diagram parsing relies on the effective detection of both textual entities (blobs) and the relationships paired with their head locations. Without both the relationship bounding box and its relationship head our algorithm does not have sufficient information to make a conclusion on a triplet set (entity-relationship-entity) associated with a given relationship.

Dataset	Blobs Detected	Relationships Detected	DP Accuracy
GN-DP-Easy	100.0%	63.47%	51.43%
GN-DP-Difficult	89.02%	27.59%	29.17%

Table 5.3. GeneNet-DP performance when dataset is separated subjectively into an equal distribution of easy and difficult diagrams.

Chapter 6 Conclusion

In this chapter, we will provide a collection of the limitations of our proposed system, and describe the future works planned for this research. Additionally, we will conclude the thesis with an overall discussion of the work completed.

6.1 Limitations & Future Works

In this thesis, we develop a collection of computer vision methods to parse genetic models. The goal of this parsing these diagrams is to create a textual description that could be used to recommend associated genes given a particular gene search query. Within the discussions of our two steps of transcription being region detection and diagram parsing, we offer various limitations or shortcomings which impacted the overall evaluation of real genetic models. Some of the shortcomings of our approach are the following:

- **Detecting with large blobs**: Isolating one bounding box around a large blob. Large blobs may contain multiple textual features in very close proximity (association) and textual features which have special characters separating a sequence of text.
- **Detecting long relationships**: Relationships that span across the whole image, not well represented in GeneNetSyn due to the random nature of blob and relationship placement. Long relationships can also represent large relationships in real genetic models which are substantially bigger in dimensions, thus relationship/blob scaling being much larger.
- **Blobs with underlined text**: Our system has yet to be trained to understand blobs in which whose textual features have an underline, which causes it to occasionally

classify an underline paired with a character to be a relationship.

- **Text above relationships**: Although considered two different object classes, our system and datasets were not developed to understand a middle textual feature above a relationship. Often being smaller fonts tight to the relationship line, our system often mistakes text above relationships as part of the relationship visual characteristic.
- **Very small or large genetic models**: Throughout this thesis we propose training a collection of computer vision methods on imagery that has a 512 width and height. When evaluated on genetic models which are much larger or smaller than the aforementioned size, our region detection network may struggle to detect all present object classes.
- **Multiple close relationships**: When multiple relationships within a genetic model are in close proximity, the active contour model can occasionally be attracted to another relationship. This will cause a false positive as although a relationship has been isolated, it may not correctly pair with the two textual features it belongs to.

In biology, there is a wide amount of other uses for genetic models other than conveying activation and inhibition relationships. An extremely important factor for the work in this thesis to advance in future works is the use of feedback from domain experts in the field of biology to indicate other interactions present in genetic models. The following additional functions and meanings used in genetic models are considered future work, as we continue to iterate to improve the capabilities of our systems.

- When multiple genes are close to each other (often connected visually with polygons), we can infer that there is an association or a bonding present.
- Relationships within genetic models span more than simply activation and inhibition relationships. A couple of examples of other relationships include association and textual features above or below relationship lines indicating further interaction.
- Within a particular genetic model, if a majority of the relationships are solid lines

and some dashed lines are present this can indicate multiple meanings. These additional meanings may be of interest to biological researchers to know which relationships between genes have this different style of the relationship line.

- In some relationships within a genetic model, there may be multiple different relationship lines going from one gene “Gene A” towards “Gene B”. This often has further meaning biological researchers may be interested in.

6.2 Discussion

Throughout this thesis, we outline the problem of generating a textual description of the underlying relationships between constituents in a biological diagram. We introduce multiple contributions for applying computer vision to visual illustrations, more specifically publication figures which are genetic models. Furthermore, we introduce a collection of both synthetic and real genetic model datasets to encourage a paradigm shift towards the next generation of bioinformatics tools.

We propose to solve the task of diagram parsing. Using an assortment of computer vision applications we successfully go from a given figure from a PubMed citation to a parsed genetic model. To be capable of training modern computer vision approaches that include the use of deep neural networks, we require large amounts of richly annotated data. Within the research field of computer vision, there is a wide collection of natural image datasets which offer ground truth for many specific applications. A dataset of this nature does not exist in the field of bioinformatics due to the work outlined being the first work of its kind. We introduce a collection of datasets containing both synthetic and real genetic models, GeneNetSyn and GeneNet respectively. The three pillars of computer vision applications are required to fulfill our proposed solution included: genetic model classification, region detection, and diagram parsing.

To classify a publication figure, we propose a Triplet classification network to determine whether a given figure is a genetic model, thus can be subsequently parsed. To improve our study, we propose GeneModelNet a feature extraction network instantiated with uniform weights, and compare it to pre-trained popular feature extraction networks used on natural imagery. We conclude that VGG19 to be the best performing feature extraction network to pass a set of features to the Triplet classification network. VGG19 successfully obtaining 95% and 93.07% classification accuracy on GeneNet-98 and GeneNet-500 respectfully.

Once a given diagram is determined to be a genetic model, we are interested in gathering the regions of interest to be the initial input to our probabilistic diagram parsing approach. With the use of transfer learning and domain adaptation, a YOLOv5m network is trained on our most complex set of synthetic diagrams GeneNetSyn Version 3. Lacking a large dataset of richly annotated real genetic models, we find it suitable to train the object detection network on synthetic genetic models. These synthetic genetic models are created to best mimic the visual characteristics of a real genetic model. A few shortcomings were found when transitioning our trained network to real genetic models. This predominately can be attributed to the immense visual differences from image to image caused by citation creators adding artistic features to their figures.

Finally, we employ an algorithm for diagram parsing leveraging active contour models. Active contour models remove the need to train a deep neural network on annotated arrows and relationship lines, hypothetically leveraging its capability to generalize on any relationship line style. The initial snake of the active contour model is a set based on the detected region of the relationships bounding box in a diagonal pattern from left-to-right and right-to-left. Once the best snake is accepted and the respective relationship head is found, entity-relationship-entity triplet sets are developed by selecting the two closest textual features to the tip and tail of the relationship. Although obtaining approximately 50% relationship accuracy in the quantitative evaluation, the shortcomings and success of the aforementioned approach are discussed qualitatively to validate its success.

Diagram parsing is a complex task that requires various applications of computer vision to work coherently to effectively create a textual description of a biological diagram. Our approach to diagram parsing although not obtaining human-level accuracy lays the foundation of a new interdisciplinary field within computer vision. A solution is not only proposed, but we introduce a large set of datasets that can be used to both train and evaluate any future work to improve this space.

References

[1] Z. Wang, Z. Yang, and F. Li, “Updates on molecular mechanisms in the development of branched trichome in arabidopsis and non-branched in cotton,” *Plant Biotechnology Journal*, vol. 17, 05 2019.

[2] A. Kembhavi, M. Salvato, E. Kolve, M. J. Seo, H. Hajishirzi, and A. Farhadi, “A diagram is worth a dozen images,” in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), vol. 9908 of *Lecture Notes in Computer Science*, pp. 235–251, Springer, 2016.

[3] A. Krizhevsky, “One weird trick for parallelizing convolutional neural networks,” *CoRR*, vol. abs/1404.5997, 2014.

[4] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: common objects in context,” in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V* (D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), vol. 8693 of *Lecture Notes in Computer Science*, pp. 740–755, Springer, 2014.

[5] Y. Ganin and V. S. Lempitsky, “Unsupervised domain adaptation by backpropagation,” in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015* (F. R. Bach and D. M. Blei, eds.), vol. 37 of *JMLR Workshop and Conference Proceedings*, pp. 1180–1189, JMLR.org, 2015.

[6] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, “Domain adaptive faster r-cnn for object detection in the wild,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[7] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *International Journal of Computer Vision*, vol. 1, pp. 321–331, 1988.

[8] R. H. Bloomer and C. Dean, “Fine-tuning timing: natural variation informs the mechanistic basis of the switch to flowering in Arabidopsis thaliana,” *Journal of Experimental Botany*, vol. 68, pp. 5439–5452, 08 2017.

[9] P. Carbonero, R. Iglesias-Fernández, and J. Vicente-Carbajosa, “The AFL subfamily of B3 transcription factors: evolution and function in angiosperm seeds,” *Journal of Experimental Botany*, vol. 68, pp. 871–880, 12 2016.

[10] F.-Y. Hung, F.-F. Chen, C. Li, C. Chen, J.-H. Chen, Y. Cui, and K. Wu, “The ldl1/2-hda6 histone modification complex interacts with toc1 and regulates the core circadian clock components in arabidopsis,” *Frontiers in Plant Science*, vol. 10, p. 233, 2019.

[11] S.-I. Inoue and T. Kinoshita, “Blue light regulation of stomatal opening and the plasma membrane h(+)-atpase,” *Plant physiology*, vol. 174, pp. 531–538, Jun 2017. 28465463[pmid].

[12] P. Robles and V. Quesada, “Transcriptional and post-transcriptional regulation of organellar gene expression (oge) and its roles in plant salt tolerance,” *International Journal of Molecular Sciences*, vol. 20, no. 5, 2019.

[13] D. Osuna, P. Prieto, and M. Aguilar, “Control of seed germination and plant development by carbon and nitrogen availability,” *Frontiers in Plant Science*, vol. 6, 11 2015.

[14] H. W. Lee, C. Cho, S. K. Pandey, Y. Park, M.-J. Kim, and J. Kim, “Lbd16 and lbd18 acting downstream of arf7 and arf19 are involved in adventitious root formation in arabidopsis,” *BMC Plant Biology*, vol. 19, p. 46, Jan 2019.

[15] S. Ortuño-Miquel, E. Rodríguez-Cazorla, E. A. Zavala-Gonzalez, A. Martínez-Laborda, and A. Vera, “Arabidopsis hua enhancer 4 delays flowering by upregulating the mads-box repressor genes flc and maf4,” *Scientific Reports*, vol. 9, p. 1478, Feb 2019.

[16] S.-V. Schiessl, E. Katche, E. Ihien, H. S. Chawla, and A. S. Mason, “The role of genomic structural variation in the genetic improvement of polyploid crops,” *The Crop Journal*, vol. 7, no. 2, pp. 127–140, 2019.

[17] S.-Y. Dai, W.-H. Hsu, and C.-H. Yang, “The gene anther dehiscence repressor (adr) controls male fertility by suppressing the ros accumulation and anther cell wall thickening in arabidopsis,” *Scientific reports*, vol. 9, pp. 5112–5112, Mar 2019. 30911018[pmid].

[18] W. Ge and C. M. Steber, “Positive and negative regulation of seed germination by the arabidopsis ga hormone receptors, gid1a, b, and c,” *Plant direct*, vol. 2, p. e00083, September 2018.

[19] Z. Xie, T. M. Nolan, H. Jiang, and Y. Yin, “Ap2/erf transcription factor regulatory networks in hormone and abiotic stress responses in arabidopsis,” *Frontiers in Plant Science*, vol. 10, p. 228, 2019.

[20] K. Chapman, M. Taleski, H. Ogilvie, N. Imin, and M. Djordjevic, “Cep-cepr1 signalling inhibits the sucrose-dependent enhancement of lateral root growth,” *Journal of experimental botany*, vol. 70, pp. 3955–3967, 08 2019.

[21] I. Chérel and I. Gaillard, “The complex fine-tuning of k+ fluxes in plants in relation to osmotic and ionic abiotic stresses,” *International Journal of Molecular Sciences*, vol. 20, no. 3, 2019.

[22] J. Glenn, S. Alex, B. Jirka, C. Liu, H. Adam, D. Laurentiu, P. Jake, Y. Lijun, R. Prashant, and R. Ferriday, “ultralytics/yolov5: v3.0,” Aug. 2020.

[23] Y. Watanabe and M. Nagao, “Diagram understanding using integration of layout information and textual information,” in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, (USA), p. 1374–1380, Association for Computational Linguistics, 1998.

[24] L. O’Gorman and R. Kasturi, “Executive briefing : document image analysis,” 1997.

[25] K. D. Ferguson, R. W. Forbus, *Advances in Analogy Research*, ch. Telling juxtapositions: Using repetition and alignable difference in diagram understanding, pp. 109–117. New Bulgarian University, 1998.

[26] R. K. Srihari, "Computational models for integrating linguistic and visual information: A survey," *Artificial Intelligence Review*, vol. 8, no. 5, pp. 349–369, 1994.

[27] N. Provart, "Summary of arabidopsis bioinformatic survey circulated to the masc and naasc mailing lists, and bioanalytic resource facebook and twitter accounts," 2018.

[28] J. Waese, J. Fan, A. Pasha, H. Yu, G. Fucile, R. Shi, M. Cumming, L. A. Kelley, M. J. Sternberg, V. Krishnakumar, E. Ferlanti, J. Miller, C. Town, W. Stuerzlinger, and N. J. Provart, "ePlant: Visualizing and Exploring Multiple Levels of Data for Hypothesis Generation in Plant Biology," *The Plant Cell*, vol. 29, pp. 1806–1821, 08 2017.

[29] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, "Cross-domain weakly-supervised object detection through progressive domain adaptation," *CoRR*, vol. abs/1803.11365, 2018.

[30] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[31] D. Kim, Y. Yoo, J. Kim, S. Lee, and N. Kwak, "Dynamic graph generation network: Generating relational knowledge from diagrams," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 4167–4175, Computer Vision Foundation / IEEE Computer Society, 2018.

[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," 2016. To appear.

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998.

[34] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, 2017.

[35] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014.

[36] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *International Journal of Computer Vision*, vol. 123, pp. 32–73, May 2017.

[37] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016.

[38] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018.

[39] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020.

[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "Imagenet: A large-scale hierarchical image database.," in *CVPR*, pp. 248–255, IEEE Computer Society, 2009.

[41] A. Krizhevsky, "Learning multiple layers of features from tiny images," pp. 32–33, 2009.

[42] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013.

[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 2818–2826, IEEE Computer Society, 2016.

[45] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778, IEEE Computer Society, 2016.

[46] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$1mb model size,” *CoRR*, vol. abs/1602.07360, 2016.

[47] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2261–2269, IEEE Computer Society, 2017.

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” *CoRR*, vol. abs/1912.01703, 2019.

[49] G. Koch, R. Zemel, and R. Salakhutdinov, “Siamese neural networks for one-shot image recognition,” 2015.

[50] E. Hoffer and N. Ailon, “Deep metric learning using triplet network,” in *Similarity-Based Pattern Recognition - Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015, Proceedings* (A. Feragen, M. Pelillo, and M. Loog, eds.), vol. 9370 of *Lecture Notes in Computer Science*, pp. 84–92, Springer, 2015.

[51] L. Duan, I. Tsang, and D. Xu, “Domain transfer multiple kernel learning,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, pp. 1 – 1, 05 2011.

[52] M. Long, Y. Cao, J. Wang, and M. Jordan, “Learning transferable features with deep adaptation networks,” in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 97–105, PMLR, 07–09 Jul 2015.

[53] P. P. Busto and J. Gall, “Open set domain adaptation,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 754–763, 2017.

[54] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto, “Unified deep supervised domain adaptation and generalization,” *CoRR*, vol. abs/1709.10190, 2017.

[55] “Active contour model.” `https://scikit-image.org/docs/dev/auto_examples/edges/plot_active_contours.html`. Accessed: 2021-10-15.

[56] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” *CoRR*, vol. abs/1808.01974, 2018.

[57] A. I. Baba, N. Andrási, I. Valkai, T. Gorcsa, L. Koczka, Z. Darula, K. F. Medzihradszky, L. Szabados, A. Fehér, G. Rigó, and Cséplő, “Atcrk5 protein kinase exhibits a regulatory role in hypocotyl hook development during skotomorphogenesis,” *International Journal of Molecular Sciences*, vol. 20, no. 14, 2019.

Chapter A Tools

In this section of the appendix the tools and software suites used throughout the aforementioned system are explained. Throughout this thesis, most of the programming completed was programmed in the language Python. Furthermore, various popular Python libraries which helped bring particular functionality include OpenCV for image loading and manipulation, PIL for image rendering, and NumPy for scientific mathematics. PyTorch introduces a complete suite for most deep learning tasks, it was solely used for the development of deep neural networks throughout the research. To develop synthetic diagrams as discussed in Chapter 3.2, we use PyGame to draw our visual illustrations. When extracting textual features from real biological diagrams as shown in Chapter 3.1, we use Google Cloud Vision API optical character recognition.

A.1 PyTorch

PyTorch [48] is a Python scientific computing package, used for experiments in Chapters 4, 5.1, and 5.2. PyTorch offers a very *Pythonic* way to build neural network models effortlessly, offering various modules such as nn, optim, Dataset, and Dataloader. These modules are the foundation for building neural networks. This deep learning library offers a great amount of flexibility within these modules and high computing speed. The torch.nn module offers many pre-built neural network layers which only require a definition of setting the input and output tensor size, as well as many optional parameters. Numerous loss functions are also available in the torch.nn package, including Binary Cross-Entropy and Triplet Loss, used throughout the diagram understanding pipeline.

There are two key features that make PyTorch stand out as the future front runner of deep learning libraries compared to TensorFlow and Keras. Firstly, PyTorch offers a torch package which is a replacement for common scientific computing packages such as NumPy and SciPy with the benefit of GPU-based computation. This allows n-dimensional torch tensors, similar to NumPy ndarrays to be run as CUDA data items compared to being detached and transferred to cpu memory. Secondly, backward propagation is considered one of the most tedious parts of deep learning libraries, this is why PyTorch offers automatic differentiation and backward propagation while training models.

A.1.1 Datasets & Pre-processing

During the production of a deep neural network approach, the dataset and effective pre-processing before the development of network architecture. This is required to ensure the validity and input structure of data items. PyTorch by default supplies a wide range of pre-loaded datasets which include many of the popular natural image datasets discussed in Chapter 2.2.1. Unfortunately, there is no visual illustration or biological diagram datasets available in the torch dataset library so custom data loaders are required.

When developing custom PyTorch datasets and data loaders it is imperative to understand the input and output of the data loader, specifically a common mistake being loading imagery as a BGR image using OpenCV instead of RGB. Throughout this thesis, some of the input data to various PyTorch data loaders include RGB images, grayscale images of region detection results, bounding box ground truth, classification ground truth, and relationship triple entities for diagram understanding.

One shortcoming of developing custom data loaders is often maintaining a reasonable time complexity when loading each batch of data items which can drastically increase a neural networks training jobs running time. Pre-processing data item tensors is essential when training deep neural networks to reduce the amount of time data loaders take reading off the disk. Alternatively, another technique used in addition to pre-processing tensors is the use of caching to store images. The only issue when using caching for data items is when storing tensors larger than typical RGB images, the memory of the system can quickly run out of capacity. To ensure the most time is saved when loading data, pre-processing can be done if a particular subsequent task in a pipeline requires data processed by a deep neural network. Within this thesis, pre-processing is used to effectively

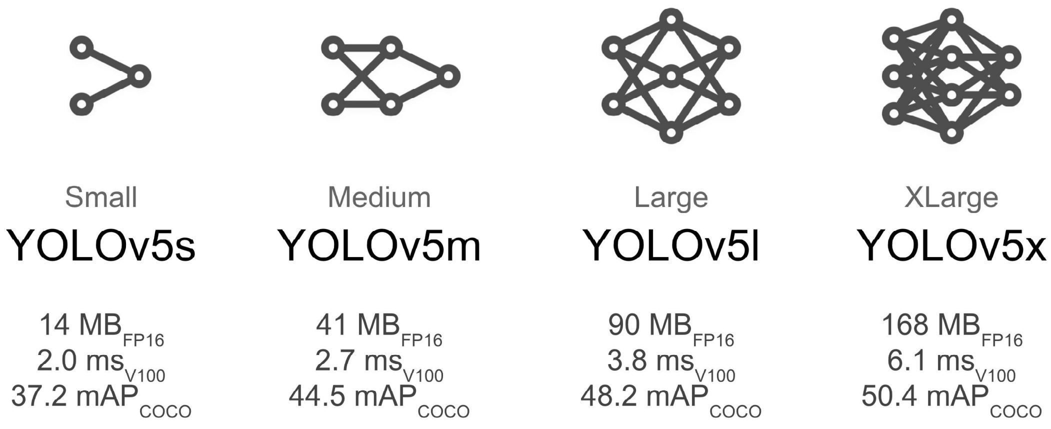


Figure A.1. Versions of YOLOv5 architecture outlined by Ultralytics [22]. Courtesy: Glenn *et al.*

load data for both contribution in Chapter 5.

A.1.2 Ultralytics YOLOv5 Implementation

Founded in 2014, Ultralytics is an open-source research company whose interested in advancing computer vision applications. Specifically, Ultralytics is well known for its implementation of a family of YOLOv5 architectures in PyTorch [48]. These implementations offer ease-free deployment and various easily accessible features for applying transfer learning and visualization of performance. They offer multiple variations of the object detection network including YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x which increase in parameters respectively. These variations of the YOLOv5 architecture are shown in Figure A.1.

In this thesis, we use the open-source implementation of YOLOv5m to train a region detection system for biological diagrams as discussed in Section 5.1. YOLOv5m was selected as it offers a good balance between parameters and space complexity. These implementations were trained on Microsoft COCO [4], meaning various learning techniques are required to adapt the pre-trained weights to biological diagrams. Furthermore, the suite offered by Ultralytics also ensures that re-training these COCO weights are simple to encourage researchers to explore object detection on a wide range of applications.

When training these networks on custom data, they are required to follow particular guidelines which will be explained below.

System YAML Files

Recently, YAML files have been found used much more than JSON files for network configuration. The YOLOv5 architectures implemented by Ultralytics use YAML files to:

- Identify the location of the dataset being used.
- Specify the number of classes being trained.
- List the names of classes in the dataset.
- Identify each piece of the network's neural architecture.

When dealing with custom datasets, it is required that the YAML file specifying data details is changed. In our implementation of biological diagrams, we use the following YAML configuration shown in Listing 1.

```
train: data/curData/images/train
val: data/curData/images/valid
test: data/curData/images/test
# nc = number of classes
nc: 5
names: ['blob', 'inhib', 'act', 'actHead', 'inHead']
```

Listing 1. YAML file named 'blobFive.yaml' used for both GeneNet and GeneNet-Syn region detection.

Data Item Formatting

Throughout most modern computer vision datasets there are various standards for how labels are saved such as comma-separated values or text files. These standards are implemented to make data loaders easy to replicate if not available. Ultralytics implementation of YOLOv5 uses the YOLO format which considers the center of a given bounding box and both it's width and height (*class, x-center, y-center, width, height*).

Directory Management

Directory management with multi-folder datasets is essential to ensure organization and differentiation between training, validation, and testing subsets. Proper directory organization is as follows:

- images $\in$ train - valid - test
- labels $\in$ train - valid - test

A.2 PyGame

PyGame is a collection of Python modules originally designed for writing video games which was last updated in January of 2017. PyGame's easy function use and modularity make creating canvas drawing trivial using the Pygame.draw module. The drawing module was specifically used for creating synthetic genetic models as discussed in Chapter 3.2. Specifically, to develop synthetic genetic models, only four drawing functions were required shown in Listing 2, which were used to draw a variation of shapes, and lines.

```
pygame.draw.ellipse() #Hollow Ovals
pygame.draw.rect()    #Rectangles / Squares
pygame.draw.lines()   #Set of lines through list of points
pygame.draw.line()    #Line drawn between two points
```

Listing 2. Draw functions used from PyGame

PyGame offers a wide range of textual elements when drawing text on a canvas of a given image. An example of the single-line textual features used in GeneNetSyn is shown in Listing 3. Furthermore, we are capable of adding any font outside of the originally supported fonts by PyGame by importing a TrueType Font file (TTF). Some of the fonts used include PyGame default font, EathomaSan, Laser, Lokey, Mintmolly, and Taylorsit. These fonts were selected as they offer a variety of different visual characteristics such as boldness, roundness, and shape.

```
def writeText(text, curFont, posX, posY):
    textFont = pygame.font.Font(get_default_font(),curFont)
    textScreen = textFont.render(text, True, (0,0,0))
    textRect = textScreen.get_rect()
    textRect.center = (posX,posY)
    screen.blit(textScreen, textRect)
```

Listing 3. Example of rendering textual features on a canvas

A.3 Optical Character Recognition

The genetic models collected for datasets discussed in Section 3.1, do not have any annotated textual features after being scraped. Optical character recognition (OCR) what and where text occurs within a given image. These two features of text occurrences are extremely important when isolating relationship triple sets as discussed in Section 5.2. We use Google's Cloud Vision API for our optical character recognition needs. Vision API offers powerful pre-trained machine learning models through both REST and RPC APIs which can be called through Python scripts in our case. The API offers various computer vision features such as object detection over millions of pre-defined categories, OCR, and image metadata collection. The primary reason for the use of Vision API's OCR is most standard OCR libraries are not trained on a diverse set of fonts and typically excels in only finding text on article-like images. Genetic models require the assumption that text can be found anywhere throughout the image, making a publication style OCR system not desirable.

Once a publication figure has been determined to be a genetic model, we can execute OCR to collect textual features. Textual features are not obtained prior as Google Cloud Vision API's OCR is a paid service, and non-genetic models will not be used. We collect all OCR output for a given image and store it into a comma separated values file for ease of use in later applications. We demonstrate our use of Vision API's OCR below in Listing 4.

This OCR textual output is used in the University of Toronto's BAR General Agricultural Intelligent Agent (GAIA) to locate which images contain a particular gene from a search query. Furthermore, we can use this output to validate the location of textual features and connect detected bounding boxes with its respective text when using our diagram understanding algorithm. Once all images are processed through OCR, we evaluate the outputting CSV files and generate a large JSON file which identifies which images and where a gene occurs within our dataset. An example of the result from the search query ”abi3/abi5” can be seen in Appendix B Listing 7.

```
client = vision.ImageAnnotatorClient()
fName = os.path.abspath(path)
with io.open(fName, 'rb') as imageFile:
    content = imageFile.read()
    imageFile.__exit__()
imageFile.close()

image = vision.types.Image(content=content)
response = client.text_detection(image=image)
texts = response.text_annotations
output = []
output.append(path)
for text in texts:
    currText = []
    currText.append(text.description)
    for vertex in text.bounding_poly.vertices:
        currText.append([vertex.x, vertex.y])
    output.append(currText)
return output
```

Listing 4. Interior of detectText function using Google Cloud Vision API’s OCR.

Chapter B Data Item Listings

Full GeneNet-DP Data Item

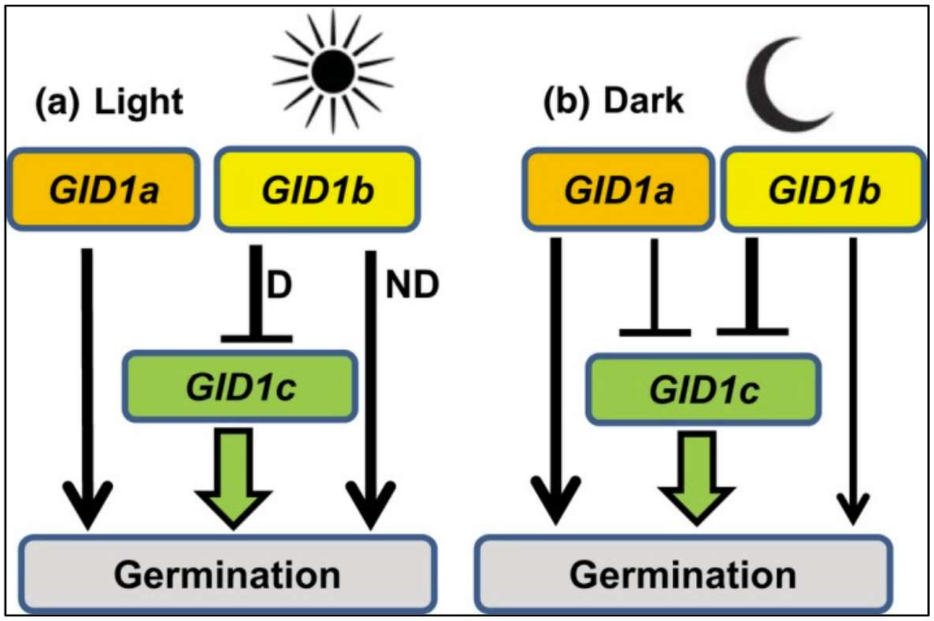


Figure B.1. Image #59 from GeneNet-500. This image was selected to be annotated in GeneNet-DP. Image shown courtesy: Steber *et al.* [18].

```
[0, 'GID1a', 1, 'Germination', 2],
[3, 'GID1b', 2, 'GID1c', 1],
[2, 'GID1c', 1, 'Germination', 2],
[3, 'GID1b', 1, 'Germination', 2],
[4, 'GID1a', 7, 'Germination', 2],
[4, 'GID1a', 6, 'GID1c', 1],
[5, 'GID1b', 6, 'GID1c', 1],
[6, 'GID1c', 7, 'Germination', 2],
[5, 'GID1b', 7, 'Germination', 2]
```

Listing 5. Image #59 annotations from relationship.csv

```
[0, 0, 'GID1a', 0.10578, 0.30021, 0.20310, 0.13034],
[0, 1, 'Germination', 0.23977, 0.93269, 0.45416, 0.12606],
[0, 2, 'GID1c', 0.253879, 0.625, 0.25387, 0.11324],
[0, 3, 'GID1b', 0.33709, 0.29914, 0.22284, 0.13675],
[0, 4, 'GID1a', 0.66431, 0.29914, 0.20310, 0.13247],
[0, 5, 'GID1b', 0.88787, 0.30021, 0.21861, 0.13461],
[0, 6, 'GID1c', 0.755994, 0.63034, 0.25105, 0.11538],
[0, 7, 'Germination', 0.73483, 0.93589, 0.44851, 0.11965],
[2, -1, 'activation', 0.08885, 0.62179, 0.06205, 0.48717],
[3, -2, 'act-head', 0.08744, 0.81837, 0.0648, 0.08974],
[1, -1, 'inhibition', 0.27221, 0.47115, 0.09026, 0.1773],
[4, -2, 'inhib-head', 0.2729, 0.53632, 0.09167, 0.04273],
[2, -1, 'activation', 0.24964, 0.77670, 0.0959, 0.18162],
[3, -2, 'act-head', 0.24541, 0.82478, 0.0959, 0.07265],
[2, -1, 'activation', 0.39633, 0.62393, 0.0648, 0.48717],
[3, -2, 'act-head', 0.39562, 0.81837, 0.0634, 0.09401],
[2, -1, 'activation', 0.59661, 0.61645, 0.0648, 0.48931],
[3, -2, 'act-head', 0.59661, 0.80982, 0.0648, 0.10256],
[1, -1, 'inhibition', 0.70239, 0.46367, 0.09026, 0.17948],
[4, -2, 'inhib-head', 0.70380, 0.52564, 0.08744, 0.04700],
[1, -1, 'inhibition', 0.81241, 0.46047, 0.09026, 0.17307],
[4, -2, 'inhib-head', 0.81100, 0.52243, 0.08180, 0.04487],
[2, -1, 'activation', 0.74894, 0.77777, 0.08744, 0.17094],
[3, -2, 'act-head', 0.75035, 0.82051, 0.08744, 0.08119],
[2, -1, 'activation', 0.91748, 0.61645, 0.04936, 0.48076],
[3, -2, 'act-head', 0.91748, 0.821581, 0.04654, 0.07051]
```

Listing 6. Image #59 annotations from relaData.csv

Textual Occurences of abi3/abi5 in GeneNet

```
"abi3/abi5":
[
    {"imageName": "/tpc1902454f09.jpg",
    "bbox": [[293, 173], [388, 173],
             [388, 186], [293, 186]]},
    {"imageName": "/nihms892484f3.jpg",
    "bbox": [[186, 611], [232, 611],
             [232, 618], [186, 618]]}
]
```

Listing 7. JSON data item for 'abi3/abi5'

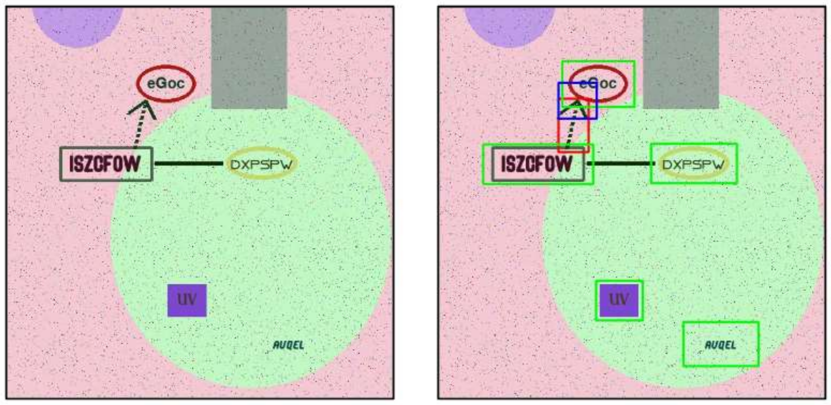


Figure B.2. Example testing image #11041 from Variable Heads. The original image (left) and ground truth image (right).

Full GeneNetSyn Data Item

To demonstrate the availability of data for a given data item, we will show all of the information for an image in the variable heads dataset of GeneNetSyn. We first begin with the actual synthetic image shown in Figure B.2. Image #11401 is quite a sparse image containing: 5 blobs, 1 activation relationship, and 1 false relationship line. This image contains blobs with and without dependent backgrounds for their textual features to challenge neural networks to only isolate text and not identify shapes as blobs. In particular, a background noise shape (lime green circle) is present which would attempt to confuse a given object detection network as 3 blobs can be found within this region.

Enabling these diagrams to be used as a training resource for object detection networks, we generate the Darknet labels shown in Table B.1. To ensure both standard and normalized variations of the coordinates for bounding boxes was given, we have broken the data item into three comma separated value (CSV) files for ease of access and readability.

The entry of the CSV below primarily contains the standardized values for the bounding boxes. The first line of the entry contains the number of blobs followed by the number of relationships present in the image. The second line of the entry contains the standardized values of the blob positions (Center X, Center Y, Width, Height). Following

Class #	Center X	Center Y	Width	Height
0	0.466796875	0.75	0.1015625	0.083984375
0	0.66015625	0.3984375	0.185546875	0.0859375
0	0.412109375	0.1953125	0.15625	0.09765625
0	0.255859375	0.400390625	0.23828125	0.0859375
0	0.73046875	0.861328125	0.1640625	0.09375
2	0.34765625	0.30078125	0.078125	0.13671875
3	0.359375	0.23828125	0.109375	0.1015625

Table B.1. Darknet labels of image #11041.

this, the actual points of each blobs bounding box are given.

```
[5, 1]
[[239, 384, 52, 43], [338, 204, 95, 44], [211, 100, 80, 50],
    [131, 205, 122, 44], [374, 441, 84, 48]]
[[[208, 358], [270, 358], [208, 410], [270, 410]]
[[281, 178], [395, 178], [281, 230], [395, 230]]
[[163, 70], [259, 70], [163, 130], [259, 130]]
[[58, 179], [204, 179], [58, 231], [204, 231]]
[[324, 412], [424, 412], [324, 470], [424, 470]]]
[[2, [[158, 119], [198, 119], [158, 189], [198, 189]]]]
```

Listing 8. Entry of 11041 from SyntheticData.csv

The entry of the CSV below contains the darknet labels shown in Table B.1, and additionally labelling the textual feature attatched to the blob, or what relationship / head type the region contains. Furthermore in the second column of the entries, the blobs are indexed based on their generation pattern.

```
[[0, 0, 'UV', 0.466796875, 0.75, 0.1015625, 0.083984375],
[0, 1, 'dXpspW', 0.66015625, 0.3984375, 0.185546875, 0.0859375],
[0, 2, 'eGoc', 0.412109375, 0.1953125, 0.15625, 0.09765625],
[0, 3, 'isZcfOW', 0.255859375, 0.400390625, 0.23828125, 0.0859375],
[0, 5, 'auqEL', 0.73046875, 0.861328125, 0.1640625, 0.09375],
[2, −1, 'activation', 0.34765625, 0.30078125, 0.078125, 0.13671875],
[3, −2, 'act−head', 0.359375, 0.23828125, 0.109375, 0.1015625]]"
```

Listing 9. Entry of 11041 from RelaData.csv

The entry of the CSV below contains relationship information present in the image. In particular this entry relies on RelaData.csv shown above to reference the index to identify which blob is within the relationship. Each cell is generated as: To Blob Index (tail), To Blob Text, From Blob Index (tip), From Blob Text, and Relationship Type.

```
[3, 'isZcfOW', 2, 'eGoc', 2]
```

Listing 10. Entry of 11041 from Relationships.csv